

Z80 CPU - INSTRUCTION SET

Z80

SGS  ATES

Z80TM CPU - INSTRUCTION SET



quantum electronics

Box 391262

Bramley

2018

This book on the instruction set of the Z80 CPU is not intended exclusively as an application support for the device itself but forms the second part of the instruction manual for SGS-ATES CLZ80 microcomputer, which is based on the Z80 microprocessor. The paragraphs are therefore numbered accordingly.

NOTE: Execution time (E.T.) for each instruction is given in microseconds for an assumed 4 MHZ clock. Total machine cycles (M) are indicated with total clock periods (T States). Also indicated are the number of T States for each M cycle. For example:

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

indicates that the instruction consists of 2 machine cycles. The first cycle contains 4 clock periods (T States). The second cycle contains 3 clock periods for a total of 7 clock periods or T States. The instruction will execute in 1.75 microseconds.

Register format is shown for each instruction with the most significant bit to the left and the least significant bit to the right.

Z 80 INSTRUCTION SET TABLE OF CONTENTS

	PAGE	SECTION
— 8 BIT LOAD GROUP	5	2.1
— 16 BIT LOAD GROUP.	31	2.2
— EXCHANGE, BLOCK TRANSFER AND SEARCH GROUP	55	2.3
— 8 BIT ARITHMETIC AND LOGICAL GROUP	79	2.4
— GENERAL PURPOSE ARITHMETIC AND CPU CONTROL GROUPS	111	2.5
— 16 BIT ARITHMETIC GROUP	127	2.6
— ROTATE AND SHIFT GROUP	145	2.7
— BIT SET, RESET AND TEST GROUP	185	2.8
— JUMP GROUP	203	2.9
— CALL AND RETURN GROUP	221	2.10
— INPUT AND OUTPUT GROUP	237	2.11
— FLAGS	261	2.12
— INDEX.	267	



quantum electronics

Box 391262

Bramley

2018

2.1 - 8 BIT LOAD GROUP -

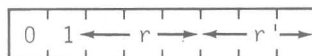
2.1.1

LD r, r'

Operation: $r \leftarrow r'$

Format:

<u>Opcode</u>	<u>Operands</u>
LD	r, r'



Description:

The contents of any register r' are loaded into any other register r . Note: r, r' identifies any of the registers A, B, C, D, E, H, or L, assembled as follows in the object code:

<u>Register</u>	<u>r, r'</u>
A	= 111
B	= 000
C	= 001
D	= 010
E	= 011
H	= 100
L	= 101

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.0

Condition Bits Affected: None

Example:

If the H register contains the number 8AH, and the E register contains 10H, the instruction

LD H, E

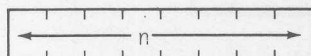
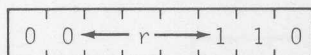
would result in both registers containing 10H.

2.1.2

LD r, n

Operation: $r \leftarrow n$ Format:

<u>Opcode</u>	<u>Operands</u>
LD	r, n

Description:

The eight-bit integer n is loaded into any register r , where r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register r

A = 111
 B = 000
 C = 001
 D = 010
 E = 011
 H = 100
 L = 101

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: NoneExample:

After the execution of

LD E, A5H

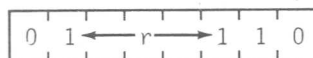
the contents of register E will be A5H.

2.1.3

LD r, (HL)

Operation: $r \leftarrow (HL)$ Format:

<u>Opcode</u>	<u>Operands</u>
LD	r, (HL)

Description:

The eight-bit contents of memory location (HL) are loaded into register r, where r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

<u>Register</u>	<u>r</u>
A	= 111
B	= 000
C	= 001
D	= 010
E	= 011
H	= 100
L	= 101

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: NoneExample:

If register pair HL contains the number 75A1H, and memory address 75A1H contains the byte 58H, the execution of

LD C, (HL)

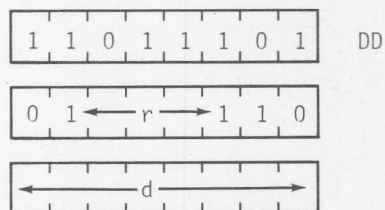
will result in 58H in register C.

2.1.4

LD r, (IX+d)

Operation: $r \leftarrow (IX+d)$ Format:

<u>Opcode</u>	<u>Operands</u>
LD	r, (IX+d)

Description:

The operand (IX+d) (the contents of the Index Register IX summed with a displacement integer d) is loaded into register r, where r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

<u>Register</u>	<u>r</u>
A	= 111
B	= 000
C	= 001
D	= 010
E	= 011
H	= 100
L	= 101

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: NoneExample:

If the Index Register IX contains the number 25AFH, the instruction

LD B, (IX+19H)

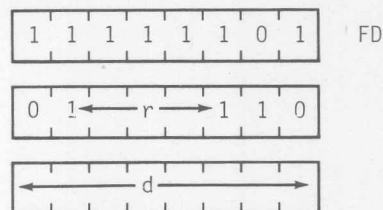
will cause the calculation of the sum $25AFH + 19H$, which points to memory location $25C8H$. If this address contains byte $39H$, the instruction will result in register B also containing $39H$.

2.1.5

LD r, (IY+d)

Operation: $r \leftarrow (IY+d)$ Format:

<u>Opcode</u>	<u>Operands</u>
LD	r, (IY+d)

Description:

The operand (IY+d) (the contents of the Index Register IY summed with a displacement integer d) is loaded into register r, where r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register r

A = 111
 B = 000
 C = 001
 D = 010
 E = 011
 H = 100
 L = 101

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

If the Index Register IY contains the number 25AFH, the instruction

LD B, (IY+19H)

will cause the calculation of the sum $25AFH + 19H$, which points to memory location 25C8H. If this address contains byte 39H, the instruction will result in register B also containing 39H.

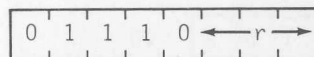
2.1.6

LD (HL), r

Operation: (HL) ← r

Format:

<u>Opcode</u>	<u>Operands</u>
LD	(HL), r



Description:

The contents of register r are loaded into the memory location specified by the contents of the HL register pair. The symbol r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register r

A = 111
 B = 000
 C = 001
 D = 010
 E = 011
 H = 100
 L = 101

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

If the contents of register pair HL specifies memory location 2146H, and the B register contains the byte 29H, after the execution of

LD (HL), B

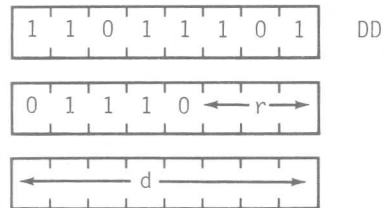
memory address 2146H will also contain 29H.

2.1.7

LD (IX+d), r

Operation: (IX+d) $\leftarrow$ rFormat:

<u>Opcode</u>	<u>Operands</u>
LD	(IX+d), r

Description:

The contents of register r are loaded into the memory address specified by the contents of Index Register IX summed with d, a two's complement displacement integer. The symbol r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register r

A = 111
 B = 000
 C = 001
 D = 010
 E = 011
 H = 100
 L = 101

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

If the C register contains the byte 1CH, and the Index Register IX contains 3100H, then the instruction

LD (IX+6H), C

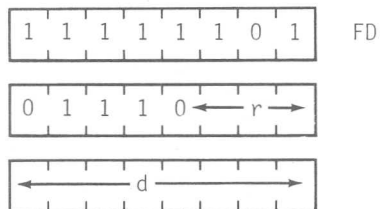
will perform the sum $3100H + 6H$ and will load 1CH into memory location 3106H.

2.1.8

LD (IY+d), r

Operation: (IY+d) ← rFormat:

<u>Opcode</u>	<u>Operands</u>
LD	(IY+d), r

Description:

The contents of register r are loaded into the memory address specified by the sum of the contents of the Index Register IY and d, a two's complement displacement integer. The symbol r is specified according to the following table.

<u>Register</u>	<u>r</u>
A	= 111
B	= 000
C	= 001
D	= 010
E	= 011
H	= 100
L	= 101

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

If the C register contains the byte 48H, and the Index Register IY contains 2A11H, then the instruction

LD (IY+4H), C

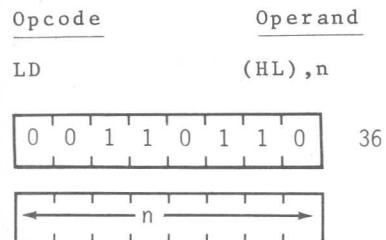
will perform the sum $2A11H + 4H$, and will load 48H into memory location 2A15.

2.1.9

LD (HL), n

Operation: (HL) ← n

Format:



Description:

Integer n is loaded into the memory address specified by the contents of the HL register pair.

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the HL register pair contains 4444H, the instruction

LD (HL), 28H

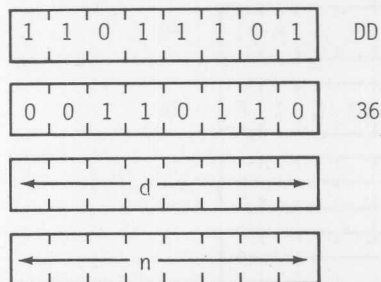
will result in the memory location 4444H containing the byte 28H.

2.1.10

LD (IX+d), n

Operation: (IX+d) $\leftarrow$ nFormat:

<u>Opcode</u>	<u>Operands</u>
LD	(IX+d), n

Description:

The n operand is loaded into the memory address specified by the sum of the contents of the Index Register IX and the two's complement displacement operand d.

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: NoneExample:

If the Index Register IX contains the number 219AH the instruction

LD (IX+5H), 5AH

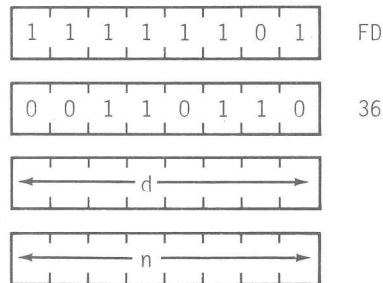
would result in the byte 5AH in the memory address 219FH.

2.1.11

LD (IY+d), n

Operation: (IY+d) ← nFormat:

<u>Opcode</u>	<u>Operands</u>
LD	(IY+d), n

Description:

Integer n is loaded into the memory location specified by the contents of the Index Register summed with a displacement integer d.

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: NONE

Example:

If the Index Register IY contains the number A940H, the instruction

LD (IY+10H), 97H

would result in byte 97 in memory location A950H.

2.1.12

LD A, (BC)

Operation: $A \leftarrow (BC)$

Format:

<u>Opcode</u>	<u>Operands</u>
LD	A, (BC)

0	0	0	0	1	0	1	0
---	---	---	---	---	---	---	---

 0A

Description:

The contents of the memory location specified by the contents of the BC register pair are loaded into the Accumulator.

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

If the BC register pair contains the number 4747H, and memory address 4747H contains the byte 12H, then the instruction

LD A, (BC)

will result in byte 12H in register A.

2.1.13

LD A, (DE)

Operation: $A \leftarrow (DE)$

Format:

<u>Opcode</u>	<u>Operands</u>
LD	A, (DE)

0	0	0	1	1	0	1	0
---	---	---	---	---	---	---	---

 1A

Description:

The contents of the memory location specified by the register pair DE are loaded into the Accumulator.

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

If the DE register pair contains the number 30A2H and memory address 30A2H contains the byte 22H, then the instruction

LD A, (DE)

will result in byte 22H in register A.

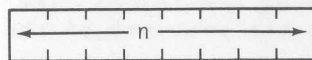
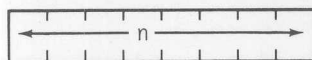
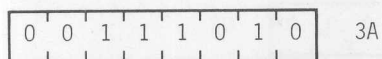
2.1.14

LD A, (nn)

Operation: $A \leftarrow (nn)$

Format:

<u>Opcode</u>	<u>Operands</u>
LD	A, (nn)



Description:

The contents of the memory location specified by the operands nn are loaded into the Accumulator. The first n operand is the low order byte of a two-byte memory address.

M CYCLES: 4 T STATES: 13(4,3,3,3) 4 MHZ E.T.: 3.25

Condition Bits Affected: None

Example:

If the contents of nn is number 8832H, and the content of memory address 8832H is byte 04H, after the instruction

LD A, (nn)

byte 04H will be in the Accumulator.

2.1.15

LD (BC), A

Operation: (BC) ← A

Format:

<u>Opcode</u>	<u>Operands</u>
LD	(BC), A

0	0	0	0	0	0	1	0	02
---	---	---	---	---	---	---	---	----

Description:

The contents of the Accumulator are loaded into the memory location specified by the contents of the register pair BC.

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

If the Accumulator contains 7AH and the BC register pair contains 1212H the instruction

LD (BC), A

will result in 7AH being in memory location 1212H.

LD (DE), A

Operation: (DE) ← A

Format:

<u>Opcode</u>	<u>Operands</u>
LD	(DE), A

0	0	0	1	0	0	1	0
---	---	---	---	---	---	---	---

12

Description:

The contents of the Accumulator are loaded into the memory location specified by the DE register pair.

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ · E.T.: 1.75

Condition Bits Affected: None

Example:

If the contents of register pair DE are 1128H, and the Accumulator contains byte A0H, the instruction

LD (DE), A

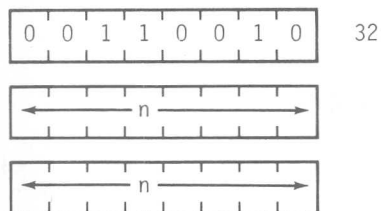
will result in A0H being in memory location 1128H.

2.1.17

LD (nn), A

Operation: (nn) ← AFormat:

<u>Opcode</u>	<u>Operands</u>
LD	(nn), A

Description:

The contents of the Accumulator are loaded into the memory address specified by the operands nn. The first n operand in the assembled object code above is the low order byte of nn.

M CYCLES: 4 T STATES: 13(4,3,3,3) 4 MHZ E.T.: 3.25

Condition Bits Affected: NoneExample:

If the contents of the Accumulator are byte D7H, after the execution of

LD (3141H), A

D7H will be in memory location 3141H.

LD A, I

Operation: $A \leftarrow I$ Format:

<u>Opcode</u>	<u>Operands</u>
LD	A, I

1	1	1	0	1	1	0	1	ED
0	1	0	1	0	1	1	1	57

Description:

The contents of the Interrupt Vector Register I are loaded into the Accumulator.

M CYCLES:2 T STATES: 9(4,5) 4 MHZ E.T.: 2.25

Condition Bits Affected:

S: Set if I-Reg. is negative;
reset otherwise
Z: Set if I-Reg. is zero;
reset otherwise
H: Reset
P/V: Contains contents of IFF2
N: Reset
C: Not affected

Example:

If the Interrupt Vector Register contains the byte 4AH, after the execution of

LD A, I

the accumulator will also contain 4AH.

2.1.19

LD A, R

Operation: $A \leftarrow R$

Format:

<u>Opcode</u>	<u>Operands</u>
LD	A,R
1 1 1 0 1 1 0 1	ED
0 1 0 1 1 1 1 1	5F

Description:

The contents of Memory Refresh Register R are loaded into the Accumulator.

M CYCLES: 2 T STATES: 9(4,5) 4 MHZ E.T.: 2.25

Condition Bits Affected:

S: Set if R-Reg. is negative;
 reset otherwise
 Z: Set if R-Reg. is zero;
 reset otherwise
 H: Reset
 P/V: Contains contents of IFF2
 N: Reset
 C: Not affected

Example:

If the Memory Refresh Register contains the byte 4AH, after the execution of

LD A,R

the Accumulator will also contain 4AH.

2.1.20

LD I, A

Operation: $I \leftarrow A$

Format:

<u>Opcode</u>	<u>Operands</u>
LD	I, A

1	1	1	0	1	1	0	1	ED
0	1	0	0	0	1	1	1	47

Description:

The contents of the Accumulator are loaded into the Interrupt Control Vector Register, I.

M CYCLES: 2 T STATES: 9(4,5) 4 MHZ E.T.: 2.25

Condition Bits Affected: None

Example:

If the Accumulator contains the number 81H, after the instruction

LD I, A

the Interrupt Vector Register will also contain 81H.

2.1.21

LD R, A

Operation: $R \leftarrow A$

Format:

<u>Opcode</u>	<u>Operands</u>								
LD	R, A								
<table border="1"><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr></table>	1	1	1	0	1	1	0	1	ED
1	1	1	0	1	1	0	1		
<table border="1"><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	0	1	0	0	1	1	1	1	4F
0	1	0	0	1	1	1	1		

Description:

The contents of the Accumulator are loaded into the Memory Refresh register R.

M CYCLES: 2 T STATES: 9(4,5) 4 MHZ E.T.: 2.25

Condition Bits Affected: None

Example:

If the Accumulator contains the number B4H, after the instruction

LD R,A

the Memory Refresh Register will also contain B4H.

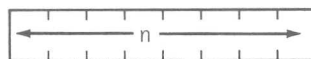
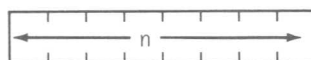
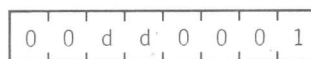
2.2 - 16 BIT LOAD GROUP -

2.2.1

LD dd, nn

Operation: dd ← nnFormat:

<u>Opcode</u>	<u>Operands</u>
LD	dd, nn

Description:

The two-byte integer nn is loaded into the dd register pair, where dd defines the BC, DE, HL, or SP register pairs, assembled as follows in the object code:

<u>Pair</u>	<u>dd</u>
BC	00
DE	01
HL	10
SP	11

The first n operand in the assembled object code is the low order byte.

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

After the execution of

LD HL, 5000H

the contents of the HL register pair will be 5000H.

2.2.2

LD IX, nn

Operation: $IX \leftarrow nn$

Format:

<u>Opcode</u>	<u>Operands</u>
---------------	-----------------

LD	IX,nn
----	-------

1 1 0 1 1 1 0 1	DD
-----------------	----

0 0 1 0 0 0 0 1	21
-----------------	----

← n →

← n →

Description:

Integer nn is loaded into the Index Register IX. The first n operand in the assembled object code above is the low order byte.

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.50

Condition Bits Affected: None

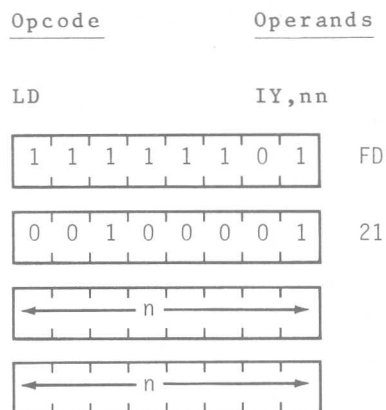
Example:

After the instruction

LD IX,45A2H

the Index Register will contain integer 45A2H.

LD IY, nn

Operation: $IY \leftarrow nn$ Format:Description:

Integer nn is loaded into the Index Register IY. The first n operand in the assembled object code above is the low order byte.

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.50

Condition Bits Affected: None

Example:

After the instruction:

LD IY,7733H

the Index Register IY will contain the integer 7733H.

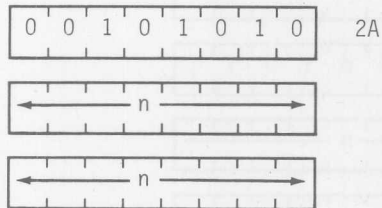
2.2.4

LD HL, (nn)

Operation: $H \leftarrow (nn+1)$, $L \leftarrow (nn)$ Format:OpcodeOperands

LD

HL, (nn)

Description:

The contents of memory address nn are loaded into the low order portion of register pair HL (register L), and the contents of the next highest memory address $nn+1$ are loaded into the high order portion of HL (register H). The first n operand in the assembled object code above is the low order byte of nn .

M CYCLES: 5 T STATES: 16(4,3,3,3,3) 4 MHZ E.T.: 4.00

Condition Bits Affected: NoneExample:

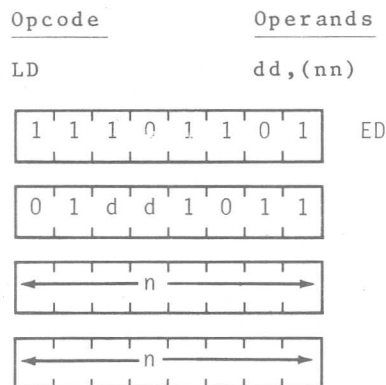
If address 4545H contains 37H and address 4546H contains A1H after the instruction

LD HL, (4545H)

the HL register pair will contain A137H.

2.2.5

LD dd, (nn)

Operation: $dd_H \leftarrow (nn+1), dd_L \leftarrow (nn)$ Format:Description:

The contents of address nn are loaded into the low order portion of register pair dd, and the contents of the next highest memory address nn+1 are loaded into the high order portion of dd. Register pair dd defines BC, DE, HL, or SP register pairs, assembled as follows in the object code:

<u>Pair</u>	<u>dd</u>
BC	00
DE	01
HL	10
SP	11

The first n operand in the assembled object code above is the low order byte of (nn).

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3) 4 MHZ E.T.: 5.00

Condition Bits Affected: None

Example:

If Address 2130H contains 65H and address 2131H contains 78H after the instruction

LD BC,(2130H)

the BC register pair will contain 7865H.

2.2.6

LD IX, (nn)

Operation: $IX_H \leftarrow (nn+1)$, $IX_L \leftarrow (nn)$ Format:

<u>Opcode</u>	<u>Operands</u>
---------------	-----------------

LD	IX, (nn)
----	----------

1 1 0 1 1 1 0 1	DD
--	----

0 0 1 0 1 0 1 0	2A
--	----

← n →

← n →

Description:

The contents of the address `nn` are loaded into the low order portion of Index Register IX, and the contents of the next highest memory address `nn+1` are loaded into the high order portion of IX. The first `n` operand in the assembled object code above is the low order byte of `nn`.

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3) 4 MHZ E.T.: 5.00

Condition Bits Affected: NoneExample:

If address 6666H contains 92H and address 6667H contains DAH, after the instruction

LD IX, (6666H)

the Index Register IX will contain DA92H.

LD IY, (nn)

Operation: $IY_H \leftarrow (nn+1)$, $IY_L \leftarrow (nn)$

Format:

<u>Opcode</u>	<u>Operands</u>
---------------	-----------------

LD	IY, (nn)
----	----------

1 1 1 1 1 1 0 1	FD
--	----

0 0 1 0 1 0 1 0	2A
--	----

← n →	
---	--

← n →	
---	--

Description:

The contents of address `nn` are loaded into the low order portion of Index Register IY, and the contents of the next highest memory address `nn+1` are loaded into the high order portion of IY. The first `n` operand in assembled object code above is the low order byte of

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3) 4 MHZ E.T.: 5

Condition Bits Affected: None

Example:

If address 6666H contains 92H and address 6667H contains DAH, after the instruction

LD IY, (6666H)

the Index Register IY will contain DA92H.

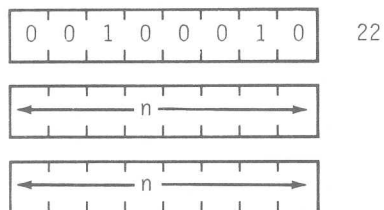
2.2.8

LD (nn), HL

Operation: (nn+1) ← H (nn) ← LFormat:

<u>Opcode</u>	<u>Operands</u>
---------------	-----------------

LD	(nn),HL
----	---------

Description:

The contents of the low order portion of register pair HL (register L) are loaded into memory address nn, and the contents of the high order portion of HL (register H) are loaded into the next highest memory address nn+1. The first n operand in the assembled object code above is the low order byte of nn.

M CYCLES: 5 T STATES: 16(4,3,3,3,3) 4 MHZ E.T.: 4.00

Condition Bits Affected: NoneExample:

If the content of register pair HL is 483AH, after the instruction

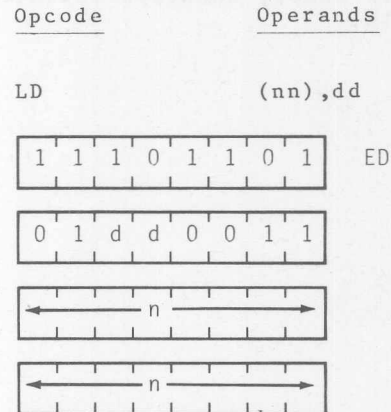
LD (B229H),HL

address B229H) will contain 3AH, and address B22AH will contain 48H.

LD (nn), dd

Operation: $(nn+1) \leftarrow dd_H, (nn) \leftarrow dd_L$

Format:



Description:

The low order byte of register pair dd is loaded into memory address nn ; the upper byte is loaded into memory address nn+1 . Register pair dd defines either BC, DE, HL, or SP, assembled as follows in the object code:

<u>Pair</u>	<u>dd</u>
BC	00
DE	01
HL	10
SP	11

The first n operand in the assembled object code is the low order byte of a two byte memory address.

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3) 4 MHZ E.T.: 5.00

Condition Bits Affected: None

Example:

If register pair BC contains the number 4644H, the instruction

LD (1000H),BC

will result in 44H in memory location 1000H, and 46H in memory location 1001H.

2.2.10

LD (nn), IX

Operation: $(nn+1) \leftarrow IX_H, (nn) \leftarrow IX_L$

Format:

<u>Opcode</u>	<u>Operands</u>
---------------	-----------------

LD	(nn), IX
----	----------

11011101	DD
--	----

00100010	22
--	----

← n →
--

← n →
--

Description:

The low order byte in Index Register IX is loaded into memory address nn ; the upper order byte is loaded into the next highest address nn+1 . The first n operand in the assembled object code above is the low order byte of nn.

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3) 4 MHZ E.T.: 5.00

Condition Bits Affected: None

Example:

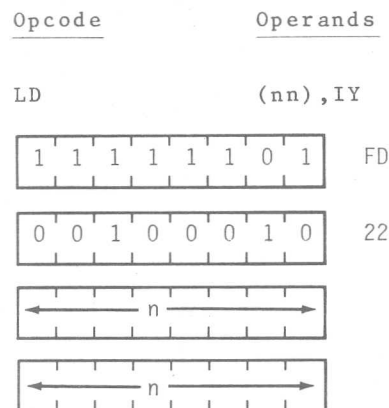
If the Index Register IX contains 5A30H, after the instruction

LD (4392H), IX

memory location 4392H will contain number 30H and location 4393H will contain 5AH.

2.2.11

LD (nn), IY

Operation: $(nn+1) \leftarrow IY_H, (nn) \leftarrow IY_L$ Format:Description:

The low order byte in Index Register IY is loaded into memory address nn ; the upper order byte is loaded into memory location nn+1 . The first n operand in the assembled object code above is the low order byte of nn.

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3) 4 MHZ E.T.: 5.00

Condition Bits Affected: NoneExample:

If the Index Register IY contains 4174H after the instruction

LD 8838H, IY

memory location 8838H will contain number 74H and memory location 8839H will contain 41H.

2.2.12

LD SP, HL

Operation: SP ← HL

Format:

<u>Opcode</u>	<u>Operands</u>
LD	SP, HL

1	1	1	1	1	0	0	1
---	---	---	---	---	---	---	---

 F9

Description:

The contents of the register pair HL are loaded into the Stack Pointer SP.

M CYCLES: 1 T STATES: 6 4 MHZ E.T.: 1.50

Condition Bits Affected: None

Example:

If the register pair HL contains 442EH, after the instruction

LD SP, HL

the Stack Pointer will also contain 442EH.

2.2.13

LD SP, IX

Operation: $SP \leftarrow IX$

Format:

<u>Opcode</u>	<u>Operands</u>
---------------	-----------------

LD	SP, IX
----	--------

1	1	0	1	1	1	0	1
---	---	---	---	---	---	---	---

DD

1	1	1	1	1	0	0	1
---	---	---	---	---	---	---	---

F9

Description:

The two byte contents of Index Register IX are loaded into the Stack Pointer SP.

M CYCLES: 2 T STATES: 10(4,6) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the contents of the Index Register IX are 98DAH, after the instruction

LD SP, IX

the contents of the Stack Pointer will also be 98DAH.

2.2.14

LD SP, IY

Operation: $SP \leftarrow IY$

Format:

<u>Opcode</u>	<u>Operands</u>
---------------	-----------------

LD	SP, IY
----	--------

1	1	1	1	1	1	0	1
---	---	---	---	---	---	---	---

 FD

1	1	1	1	1	0	0	1
---	---	---	---	---	---	---	---

 F9

Description:

The two byte contents of Index Register IY are loaded into the Stack Pointer SP.

M CYCLES: 2 T STATES: 10(4,6) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If Index Register IY contains the integer A227H, after the instruction

LD SP, IY

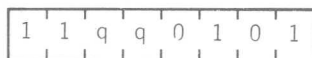
the Stack Pointer will also contain A227H.

PUSH qq

Operation: (SP-2) $\leftarrow$ qq_L, (SP-1) $\leftarrow$ qq_H

Format:

<u>Opcode</u>	<u>Operands</u>
PUSH	qq



Description:

The contents of the register pair qq are pushed into the external memory LIFO (last-in, first-out) Stack. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first decrements the SP and loads the high order byte of register pair qq into the memory address now specified by the SP; then decrements the SP again and loads the low order byte of qq into the memory location corresponding to this new address in the SP. The operand qq means register pair BC, DE, HL, or AF, assembled as follows in the object code:

<u>Pair</u>	<u>qq</u>
BC	00
DE	01
HL	10
AF	11

M CYCLES: 3 T STATES: 11(5,3,3) 4 MHZ E.T.: 2.75

Condition Bits Affected: None

Example:

If the AF register pair contains 2233H and the Stack Pointer contains 1007H, after the instruction

PUSH AF

memory address 1006H will contain 22H, memory address 1005H will contain 33H, and the Stack Pointer will contain 1005H.

PUSH IX

Operation: $(SP-2) \leftarrow IX_L, (SP-1) \leftarrow IX_H$

Format:

<u>Opcode</u>	<u>Operands</u>
PUSH	IX
1 1 0 1 1 1 0 1	DD
1 1 1 0 0 1 0 1	E5

Description:

The contents of the Index Register IX are pushed into the external memory LIFO (last-in, first-out) Stack. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first decrements the SP and loads the high order byte of IX into the memory address now specified by the SP; then decrements the SP again and loads the low order byte into the memory location corresponding to this new address in the SP.

M CYCLES: 3 T STATES: 15(4,5,3,3) 4 MHZ E.T.: 3.75

Condition Bits Affected: None

Example:

If the Index Register IX contains 2233H and the Stack Pointer contains 1007H, after the instruction

PUSH IX

memory address 1006H will contain 22H, memory address 1005H will contain 33H, and the Stack Pointer will contain 1005H.

PUSH IY

Operation: $(SP-2) \leftarrow IY_L, (SP-1) \leftarrow IY_H$

Format:

<u>Opcode</u>	<u>Operands</u>
---------------	-----------------

PUSH	IY
------	----

1 1 1 1 1 1 0 1	FD
1 1 1 0 0 1 0 1	E5

Description:

The contents of the Index Register IY are pushed into the external memory LIFO (last-in, first-out) Stack. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first decrements the SP and loads the high order byte of IY into the memory address now specified by the SP; then decrements the SP again and loads the low order byte into the memory location corresponding to this new address in the SP.

M CYCLES: 4 T STATES: 15(4,5,3,3) 4 MHZ E.T.: 3.75

Condition Bits Affected: None

Example:

If the Index Register IY contains 2233H and the Stack Pointer contains 1007H, after the instruction

PUSH IY

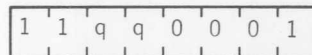
memory address 1006H will contain 22H, memory address 1005H will contain 33H, and the Stack Pointer will contain 1005H.

POP qq

Operation: $qq_H \leftarrow (SP+1), qq_L \leftarrow (SP)$

Format:

<u>Opcode</u>	<u>Operands</u>
POP	qq



Description:

The top two bytes of the external memory LIFO (last-in, first-out) Stack are popped into register pair qq. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first loads into the low order portion of qq, the byte at the memory location corresponding to the contents of SP; then SP is incremented and the contents of the corresponding adjacent memory location are loaded into the high order portion of qq and the SP is now incremented again. The operand qq defines register pair BC, DE, HL, or AF, assembled as follows in the object code:

<u>Pair</u>	<u>r</u>
BC	00
DE	01
HL	10
AF	11

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the Stack Pointer contains 1000H, memory location 1000H contains 55H, and location 1001H contains 33H, the instruction

POP HL

will result in register pair HL containing 3355H, and the Stack Pointer containing 1002H.

POP IX

Operation: $IX_H \leftarrow (SP+1), IX_L \leftarrow (SP)$

Format:

<u>Opcode</u>	<u>Operands</u>
POP	IX
1 1 0 1 1 1 0 1	DD
1 1 1 0 0 0 0 1	E1

Description:

The top two bytes of the external memory LIFO (last-in, first-out) Stack are popped into Index Register IX. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first loads into the low order portion of IX the byte at the memory location corresponding to the contents of SP; then SP is incremented and the contents of the corresponding adjacent memory location are loaded into the high order portion of IX. The SP is now incremented again.

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.50

Condition Bits Affected: None

Example:

If the Stack Pointer contains 1000H, memory location 1000H contains 55H, and location 1001H contains 33H, the instruction

POP IX

will result in Index Register IX containing 3355H, and the Stack Pointer containing 1002H.

POP IY

Operation: $IY_H \leftarrow (SP+1), IY_L \leftarrow (SP)$

Format:

<u>Opcode</u>	<u>Operands</u>
POP	IY
1 1 1 1 1 1 0 1	FD
1 1 1 0 0 0 0 1	E1

Description:

The top two bytes of the external memory LIFO (last-in, first-out) Stack are popped into Index Register IY. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first loads into the low order portion of IY the byte at the memory location corresponding to the contents of SP; then SP is incremented and the contents of the corresponding adjacent memory location are loaded into the high order portion of IY. The SP is now incremented again.

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.50

Condition Bits Affected: None

Example:

If the Stack Pointer contains 1000H, memory location 1000H contains 55H, and location 1001H contains 33H, the instruction

POP IY

will result in Index Register IY containing 3355H, and the Stack Pointer containing 1002H.

2.3 - EXCHANGE, BLOCK TRANSFER AND SEARCH GROUP -

2.3.1

EX DE, HL

Operation: DE ↔ HL

Format:

<u>Opcode</u>	<u>Operands</u>
EX	DE, HL

<table border="1" style="display: inline-table; text-align: center;"> <tr> <td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td> </tr> </table>	1	1	1	0	1	0	1	1	EB
1	1	1	0	1	0	1	1		

Description:

The two-byte contents of register pairs DE and HL are exchanged.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: None

Example:

If the content of register pair DE is the number 2822H, and the content of the register pair HL is number 499AH, after the instruction

EX DE, HL

the content of register pair DE will be 499AH and the content of register pair HL will be 2822H.

2.3.2

EX AF, AF'

Operation: AF $\leftrightarrow$ AF'Format:

<u>Opcode</u>	<u>Operands</u>
EX	AF, AF'

0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

 08
Description:

The two-byte contents of the register pairs AF and AF' are exchanged. (Note: register pair AF' consists of registers A' and F'.)

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: NoneExample:

If the content of register pair AF is number 9900H, and the content of register pair AF' is number 5944H, after the instruction

EX AF, AF'

the contents of AF will be 5944H, and the contents of AF' will be 9900H.

EXX

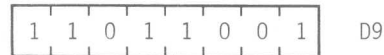
Operation: (BC) $\leftrightarrow$ (BC'), (DE) $\leftrightarrow$ (DE'), (HL) $\leftrightarrow$ (HL')

Format:

Opcode

Operands

EXX



Description:

Each two-byte value in register pairs BC, DE, and HL is exchanged with the two-byte value in BC', DE', and HL', respectively.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: None

Example:

If the contents of register pairs BC, DE, and HL are the numbers 445AH, 3DA2H, and 8859H, respectively, and the contents of register pairs BC', DE', and HL' are 0988H, 9300H, and 00E7H, respectively, after the instruction

EXX

the contents of the register pairs will be as follows:
BC: 0988H; DE: 9300H; HL: 00E7H; BC': 445AH; DE': 3DA2H;
and HL': 8859H.

2.3.4

EX (SP), HL

Operation: $H \leftrightarrow (SP+1), L \leftrightarrow (SP)$ Format:

<u>Opcode</u>	<u>Operands</u>
EX	(SP),HL

1	1	1	0	0	0	1	1	E3
---	---	---	---	---	---	---	---	----

Description:

The low order byte contained in register pair HL is exchanged with the contents of the memory address specified by the contents of register pair SP (Stack Pointer), and the high order byte of HL is exchanged with the next highest memory address (SP+1).

M CYCLES: 5 T STATES: 19(4,3,4,3,5) 4 MHZ E.T.: 4.75

Condition Bits Affected: NoneExample:

If the HL register pair contains 7012H, the SP register pair contains 8856H, the memory location 8856H contains the byte 11H, and the memory location 8857H contains the byte 22H, then the instruction

EX (SP),HL

will result in the HL register pair containing number 2211H, memory location 8856H containing the byte 12H, the memory location 8857H containing the byte 70H and the Stack Pointer containing 8856H.

EX (SP), IX

Operation: $IX_H \leftrightarrow (SP+1)$, $IX_L \leftrightarrow (SP)$

Format:

<u>Opcode</u>	<u>Operands</u>
EX	(SP), IX
1 1 0 1 1 1 0 1	DD
1 1 1 0 0 0 1 1	E3

Description:

The low order byte in Index Register IX is exchanged with the contents of the memory address specified by the contents of register pair SP (Stack Pointer), and the high order byte of IX is exchanged with the next highest memory address (SP+1).

M CYCLES: 6 T STATES: 23(4,4,3,4,3,5) 4 MHZ E.T.: 5.75

Condition Bits Affected: None

Example:

If the Index Register IX contains 3988H, the SP register pair contains 0100H, the memory location 0100H contains the byte 90H, and memory location 0101H contains byte 48H, then the instruction

EX (SP), IX

will result in the IX register pair containing number 4890H, memory location 0100H containing 88H, memory location 0101H containing 39H and the Stack Pointer containing 0100H.

EX (SP), IY

Operation: $IY_H \leftrightarrow (SP+1)$, $IY_L \leftrightarrow (SP)$

Format:

<u>Opcode</u>	<u>Operands</u>
EX	(SP), IY

1 1 1 1 1 1 0 1	FD
-----------------	----

1 1 1 0 0 0 1 1	E3
-----------------	----

Description:

The low order byte in Index Register IY is exchanged with the contents of the memory address specified by the contents of register pair SP (Stack Pointer), and the high order byte of IY is exchanged with the next highest memory address (SP+1).

M CYCLES: 6 T STATES: 23(4,4,3,4,3,5) 4 MHZ E.T.: 5.75

Condition Bits Affected: None

Example:

If the Index Register IY contains 3988H, the SP register pair contains 0100H, the memory location 0100H contains the byte 90H, and memory location 0101H contains byte 48H, then the instruction

EX (SP), IY

will result in the IY register pair containing number 4890H, memory location 0100H containing 88H, memory location 0101H containing 39H, and the Stack Pointer containing 0100H.

LDI

Operation: $(DE) \leftarrow (HL)$, $DE \leftarrow DE+1$, $HL \leftarrow HL+1$, $BC \leftarrow BC-1$

Format:

<u>Opcode</u>	<u>Operands</u>								
LDI									
<table border="1"><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr></table>	1	1	1	0	1	1	0	1	ED
1	1	1	0	1	1	0	1		
<table border="1"><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	1	0	1	0	0	0	0	0	A0
1	0	1	0	0	0	0	0		

Description:

A byte of data is transferred from the memory location addressed by the contents of the HL register pair to the memory location addressed by the contents of the DE register pair. Then both these register pairs are incremented and the BC (Byte Counter) register pair is decremented.

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S: Not affected
 Z: Not affected
 H: Reset
 P/V: Set if $BC-1 \neq 0$;
 reset otherwise
 N: Reset
 C: Not affected

Example:

If the HL register pair contains 1111H, memory location 1111H contains the byte 88H, the DE register pair contains 2222H, the memory location 2222H contains byte 66H, and the BC register pair contains 7H, then the instruction

LDI

will result in the following contents in register pairs and memory addresses:

HL	:	1112H
(1111H)	:	88H
DE	:	2223H
(2222H)	:	88H
BC	:	6H

LDIR

Operation: $(DE) \leftarrow (HL), DE \leftarrow DE+1, HL \leftarrow HL+1, BC \leftarrow BC-1$

Format:

<u>Opcode</u>	<u>Operands</u>								
LDIR									
<table><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr></table>	1	1	1	0	1	1	0	1	ED
1	1	1	0	1	1	0	1		
<table><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	1	0	1	1	0	0	0	0	BO
1	0	1	1	0	0	0	0		

Description:

This two byte instruction transfers a byte of data from the memory location addressed by the contents of the HL register pair to the memory location addressed by the DE register pair. Then both these register pairs are incremented and the BC (Byte Counter) register pair is decremented. If decrementing causes the BC to go to zero, the instruction is terminated. If BC is not zero the program counter is decremented by 2 and the instruction is repeated. Note that if BC is set to zero prior to instruction execution, the instruction will loop through 64K bytes. Also, "Interrupts will be recognized and two refresh cycles will be executed after each data transfer".

For $BC \neq 0$:

M CYCLES: 5 T STATES: 21(4,4,3,5,5) 4 MHZ E.T.: 5.25

For $BC=0$:

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Reset
N: Reset
C: Not affected

Example:

If the HL register pair contains 1111H, the DE register pair contains 2222H, the BC register pair contains 0003H, and memory locations have these contents:

(1111H) : 88H	(2222H) : 66H
(1112H) : 36H	(2223H) : 59H
(1113H) : A5H	(2224H) : C5H

then after the execution of

LDIR

the contents of register pairs and memory locations will be:

HL : 1114H
DE : 2225H
BC : 0000H

(1111H) : 88H	(2222H) : 88H
(1112H) : 36H	(2223H) : 36H
(1113H) : A5H	(2224H) : A5H

LDD

Operation: $(DE) \leftarrow (HL), DE \leftarrow DE-1, HL \leftarrow HL-1, BC \leftarrow BC-1$

Format:

<u>Opcode</u>	<u>Operands</u>
---------------	-----------------

LDD

1	1	1	0	1	1	0	1	ED
---	---	---	---	---	---	---	---	----

1	0	1	0	1	0	0	0	A8
---	---	---	---	---	---	---	---	----

Description:

This two byte instruction transfers a byte of data from the memory location addressed by the contents of the HL register pair to the memory location addressed by the contents of the DE register pair. Then both of these register pairs including the BC (Byte Counter) register pair are decremented.

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Reset
P/V:	Set if $BC-1 \neq 0$; reset otherwise
N:	Reset
C:	Not affected

Example:

If the HL register pair contains 1111H, memory location 1111H contains the byte 88H, the DE register pair contains 2222H, memory location 2222H contains byte 66H, and the BC register pair contains 7H, then the instruction

LDD

will result in the following contents in register pairs and memory addresses:

HL	:	1110H
(1111H)	:	88H
DE	:	2221H
(2222H)	:	88H
BC	:	6H

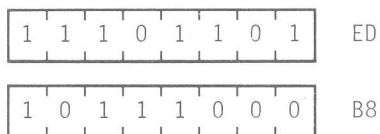
LDDR

Operation: $(DE) \leftarrow (HL)$, $DE \leftarrow DE-1$, $HL \leftarrow HL-1$, $BC \leftarrow BC-1$

Format:

<u>Opcode</u>	<u>Operands</u>
---------------	-----------------

LDDR



Description:

This two byte instruction transfers a byte of data from the memory location addressed by the contents of the HL register pair to the memory location addressed by the contents of the DE register pair. Then both of these registers as well as the BC (Byte Counter) are decremented. If decrementing causes the BC to go to zero, the instruction is terminated. If BC is not zero, the program counter is decremented by 2 and the instruction is repeated. Note that if BC is set to zero prior to instruction execution, the instruction will loop through 64K bytes. Also, "Interrupts will be recognized and two refresh cycles will be executed after each data transfer".

For $BC \neq 0$:

M CYCLES: 5 T STATES: 21(4,4,3,5,5) 4 MHZ E.T.: 5.25

For $BC = 0$:

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Reset
P/V:	Reset
N:	Reset
C:	Not affected

Example:

If the HL register pair contains 1114H, the DE register pair contains 2225H, the BC register pair contains 0003H, and memory locations have these contents:

(1114H) : A5H	(2225H) : C5H
(1113H) : 36H	(2224H) : 59H
(1112H) : 88H	(2223H) : 66H

then after the execution of

LDDR

the contents of register pairs and memory locations will be:

HL : 1111H
DE : 2222H
BC : 0000H

(1114H) : A5H	(2225H) : A5H
(1113H) : 36H	(2224H) : 36H
(1112H) : 88H	(2223H) : 88H

CPI

Operation: A - (HL), HL $\leftarrow$ HL+1, BC $\leftarrow$ BC-1

Format:

<u>Opcode</u>	<u>Operands</u>								
CPI									
<table border="1"><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr></table>	1	1	1	0	1	1	0	1	ED
1	1	1	0	1	1	0	1		
<table border="1"><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr></table>	1	0	1	0	0	0	0	1	A1
1	0	1	0	0	0	0	1		

Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. Then HL is incremented and the Byte Counter (register pair BC) is decremented.

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if A=(HL); reset otherwise
H:	Set if there is a borrow and reset otherwise.
P/V:	Set if BC-1 $\neq$ 0; reset otherwise
N:	Set
C:	Not affected

Example:

If the HL register pair contains 1111H, memory location 1111H contains 3BH, the Accumulator contains 3BH, and the Byte Counter contains 0001H, then after the execution of

CPI

the Byte Counter will contain 0000H, the HL register pair will contain 1112H, the Z flag in the F register will be set, and the P/V flag in the F register will be reset. There will be no effect on the contents of the Accumulator or address 1111H.

CPIR

Operation: A - (HL), HL $\leftarrow$ HL+1, BC $\leftarrow$ BC-1

Format:

Opcode Operands

CPIR

1	1	1	0	1	1	0	1	ED
---	---	---	---	---	---	---	---	----

1	0	1	1	0	0	0	1	B1
---	---	---	---	---	---	---	---	----

Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. The HL is incremented and the Byte Counter (register pair BC) is decremented. If decrementing causes the BC to go to zero or if A=(HL), the instruction is terminated. If BC is not zero and A $\neq$ (HL), the program counter is decremented by 2 and the instruction is repeated. Note that if BC is set to zero before instruction execution, the instruction will loop through 64K bytes, if no match is found. Also, "Interrupts will be recognized and two refresh cycles will be executed after each data transfer".

For BC $\neq$ 0 and A $\neq$ (HL):

M CYCLES: 5 T STATES: 21(4,4,3,5,5) 4 MHZ E.T.: 5.25

For BC=0 or A=(HL):

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if A=(HL);
reset otherwise
H: Set if there is a borrow
and reset otherwise.
P/V: Set if BC-1 $\neq$ 0;
reset otherwise
N: Set
C: Not affected

Example:

If the HL register pair contains 1111H, the Accumulator contains F3H, the Byte Counter contains 0007H, and memory locations have these contents:

(1111H) : 52H
(1112H) : 00H
(1113H) : F3H

then after the execution of

CPIR

the contents of register pair HL will be 1114H, the contents of the Byte Counter will be 0004H, the P/V flag in the F register will be set and the Z flag in the F register will be set.

CPD

Operation: A ← (HL); HL ← HL-1, BC ← BC-1

Format:

<u>Opcode</u>	<u>Operands</u>
CPD	

1	1	1	0	1	1	0	1
---	---	---	---	---	---	---	---

ED

1	0	1	0	1	0	0	1
---	---	---	---	---	---	---	---

A9

Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. The HL and the Byte Counter (register pair BC) are decremented.

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if A=(HL); reset otherwise
H:	Set if there is a borrow and reset otherwise.
P/V:	Set if BC-1≠0; reset otherwise
N:	Set
C:	Not Affected

Example:

If the HL register pair contains 1111H, memory location 1111H contains 3BH, the Accumulator contains 3BH, and the Byte Counter contains 0001H, then after the execution of

CPD

the Byte Counter will contain 0000H, the HL register pair will contain 1110H, the Z flag in the F register will be set, and the P/V flag in the F register will be reset. There will be no effect on the contents of the Accumulator or address 1111H.

CPDR

Operation: A - (HL); HL $\leftarrow$ HL-1, BC $\leftarrow$ BC-1

Format:

<u>Opcode</u>	<u>Operands</u>								
CPDR									
<table border="1"><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr></table>	1	1	1	0	1	1	0	1	ED
1	1	1	0	1	1	0	1		
<table border="1"><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	1	0	1	1	1	0	0	1	B9
1	0	1	1	1	0	0	1		

Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. The HL and BC (Byte Counter) register pairs are decremented. If decrementing causes the BC to go to zero or if A=(HL), the instruction is terminated. If BC is not zero and A $\neq$ (HL), the program counter is decremented by 2 and the instruction is repeated. Note that if BC is set to zero prior to instruction execution, the instruction will loop through 64K bytes, if no match is found. Also, "Interrupts will be recognized and two refresh cycles will be executed after each data transfer".

For BC $\neq$ 0 and A $\neq$ (HL):

M CYCLES: 5 T STATES: 21(4,4,3,5,5) 4 MHZ E.T.: 5.25

For BC=0 or A=(HL):

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if A=(HL);
reset otherwise
H: Set if there is a borrow
and reset otherwise.
P/V: Set if BC-1≠0;
reset otherwise
N: Set
C: Not affected

Example:

If the HL register pair contains 1118H, the Accumulator contains F3H, the Byte Counter contains 0007H, and memory locations have these contents:

(1118H) : 52H
(1117H) : 00H
(1116H) : F3H

then after the execution of

CPDR

the contents of register pair HL will be 1115H, the contents of the Byte Counter will be 0004H, the P/V flag in the F register will be set, and the Z flag in the F register will be set.

2.4. - 8 BIT ARITHMETIC AND LOGICAL GROUP -

2.4.1

ADD A, r

Operation: $A \leftarrow A + r$

Format:

<u>Opcode</u>	<u>Operands</u>
ADD	A, r



Description:

The contents of register r are added to the contents of the Accumulator, and the result is stored in the Accumulator. The symbol r identifies the registers A, B, C, D, E, H or L assembled as follows in the object code:

<u>Register</u>	<u>r</u>
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Reset
C:	Set if carry from Bit 7; reset otherwise

Example:

If the contents of the Accumulator are 44H, and the contents of register C are 11H, after the execution of

ADD A,C

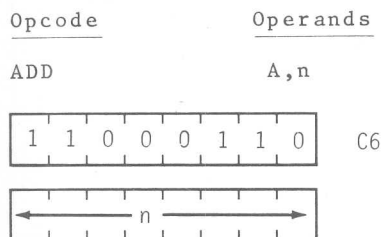
the contents of the Accumulator will be 55H.

2.4.2

ADD A, n

Operation: $A \leftarrow A + n$

Format:



Description:

The integer n is added to the contents of the Accumulator and the results are stored in the Accumulator.

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected:

S: Set if result is negative;
 reset otherwise
 Z: Set if result is zero;
 reset otherwise
 H: Set if carry from
 Bit 3; reset otherwise
 P/V: Set if overflow;
 reset otherwise
 N: Reset
 C: Set if carry from
 Bit 7; reset otherwise

Example:

If the contents of the Accumulator are 23H, after the execution of

ADD A, 33H

the contents of the Accumulator will be 56H.

ADD A, (HL)

Operation: $A \leftarrow A + (HL)$

Format:

<u>Opcode</u>	<u>Operands</u>
ADD	A, (HL)

1 0 0 0 0 1 1 0	86
-----------------	----

Description:

The byte at the memory address specified by the contents of the HL register pair is added to the contents of the Accumulator and the result is stored in the Accumulator.

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Reset
C:	Set if carry from Bit 7; reset otherwise

Example:

If the contents of the Accumulator are A0H, and the content of the register pair HL is 2323H, and memory location 2323H contains byte 08H, after the execution of

ADD A, (HL)

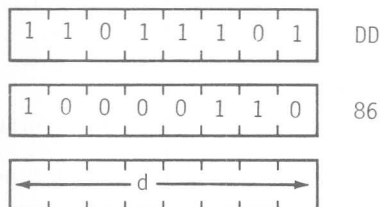
the Accumulator will contain A8H.

ADD A, (IX+d)

Operation: $A \leftarrow A + (IX+d)$

Format:

<u>Opcode</u>	<u>Operands</u>
ADD	A, (IX+d)



Description:

The contents of the Index Register (register pair IX) is added to a displacement d to point to an address in memory. The contents of this address is then added to the contents of the Accumulator and the result is stored in the Accumulator.

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Reset
C:	Set if carry from Bit 7; reset otherwise

Example:

If the Accumulator contents are 11H, the Index Register IX contains 1000H, and if the content of memory location

1005H is 22H, after the execution of

ADD A,(IX+5H)

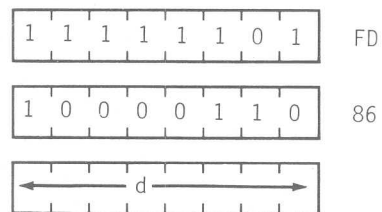
the contents of the Accumulator will be 33H.

2.4.5 ADD A, (IY+d)

Operation: $A \leftarrow A + (IY + d)$

Format:

<u>Opcode</u>	<u>Operands</u>
ADD	A, (IY+d)



Description:

The contents of the Index Register (register pair IY) is added to a displacement d to point to an address in memory. The contents of this address is then added to the contents of the Accumulator and the result is stored in the Accumulator.

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Reset
C:	Set if carry from bit 7; reset otherwise

Example:

If the Accumulator contents are 11H, the Index Register pair IY contains 1000H, and if the content of memory

location 1005H is 22H, after the execution of

ADD A,(1Y+5H)

the contents of the Accumulator will be 33H.

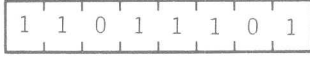
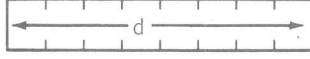
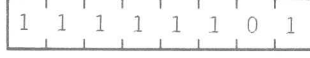
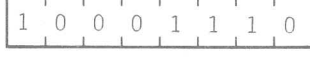
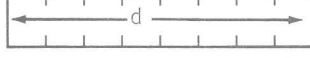
ADC A, s

Operation: $A \leftarrow A + s + CY$

Format:

<u>Opcode</u>	<u>Operands</u>
ADC	A, s

The s operand is any of r, n, (HL), (IX+d) or (IY+d) as defined for the analogous ADD instruction. These various possible opcode-operand combinations are assembled as follows in the object code:

ADC A, r		
ADC A, n		CE
		
ADC A, (HL)		8E
ADC A, (IX+d)		DD
		8E
		
ADC A, (IY+d)		FD
		8E
		

*r identifies registers B, C, D, E, H, L or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The s operand, along with the Carry Flag ("C" in the F register) is added to the contents of the Accumulator, and the result is stored in the Accumulator.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
ADC A,r	1	4	1.00
ADC A,n	2	7(4,3)	1.75
ADC A,(HL)	2	7(4,3)	1.75
ADC A,(IX+d)	5	19(4,4,3,5,3)	4.75
ADC A,(IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Set if carry from
Bit 3; reset otherwise
P/V: Set if overflow;
reset otherwise
N: Reset
C: Set if carry from
Bit 7; reset otherwise

Example:

If the Accumulator contains 16H, the Carry Flag is set, the HL register pair contains 6666H, and address 6666H contains 10H, after the execution of

ADC A,(HL)

the Accumulator will contain 27H.

SUB s

Operation: $A \leftarrow A - s$

Format:

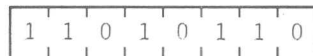
<u>Opcode</u>	<u>Operands</u>
SUB	s

The s operand is any of r,n,(HL),(IX+d) or (IY+d) as defined for the analogous ADD instruction. These various possible opcode-operand combinations are assembled as follows in the object code:

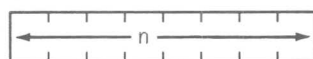
SUB r



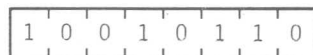
SUB n



D6

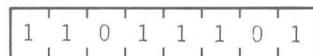


SUB (HL)

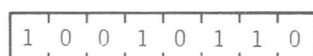


96

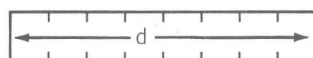
SUB (IX+d)



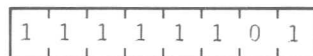
DD



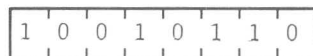
96



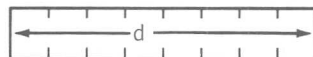
SUB (IY+d)



FD



96



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

<u>Register</u>	<u>r</u>
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The s operand is subtracted from the contents of the Accumulator, and the result is stored in the Accumulator.

<u>INSTRUCTION</u>	<u>M CYCLES</u>	<u>T STATES</u>	<u>4 MHZ E.T.</u>
SUB r	1	4	1.00
SUB n	2	7(4,3)	1.75
SUB (HL)	2	7(4,3)	1.75
SUB (IX+d)	5	19(4,4,3,5,3)	4.75
SUB (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Set if there is a borrow
and reset otherwise.
P/V: Set if overflow;
reset otherwise
N: Set
C: Set if there is a borrow
and reset otherwise.

Example:

If the Accumulator contains 29H and register D contains 11H, after the execution of

SUB D

the Accumulator will contain 18H.

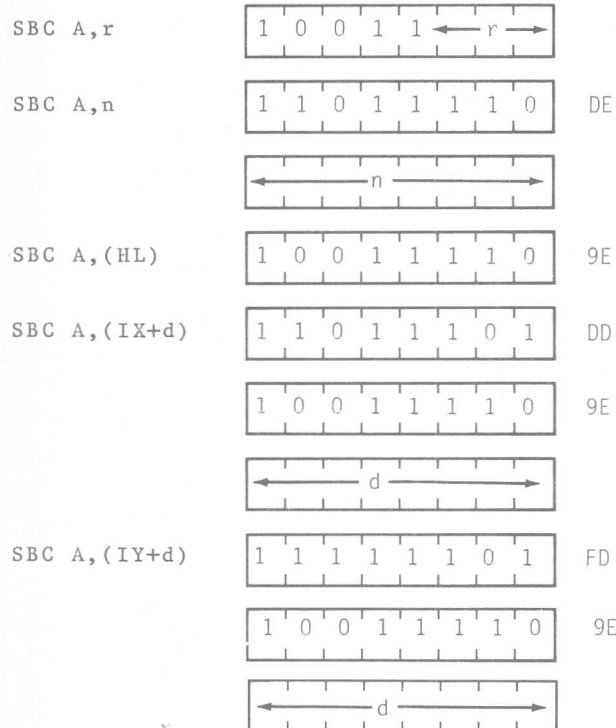
SBC A, s

Operation: $A \leftarrow A - s - CY$

Format:

<u>Opcode</u>	<u>Operands</u>
SBC	A, s

The s operand is any of r,n,(HL),(IX+d) or (IY+d) as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

<u>Register</u>	<u>r</u>
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The s operand, along with the Carry Flag ("C" in the F register) is subtracted from the contents of the Accumulator, and the result is stored in the Accumulator.

<u>INSTRUCTION</u>	<u>M CYCLES</u>	<u>T STATES</u>	<u>4 MHZ E.T.</u>
SBC A,r	1	4	1.00
SBC A,n	2	7(4,3)	1.75
SBC A,(HL)	2	7(4,3)	1.75
SBC A,(IX+d)	5	19(4,4,3,5,3)	4.75
SBC A,(IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative;
 reset otherwise
 Z: Set if result is zero;
 reset otherwise
 H: Set if there is a borrow
 and reset otherwise.
 P/V: Set if overflow;
 reset otherwise
 N: Set
 C: Set if there is a borrow
 and reset otherwise.

Example:

If the Accumulator contains 16H, the carry flag is set, the HL register pair contains 3433H, and address 3433H contains 05H, after the execution of

SBC A,(HL)

the Accumulator will contain 10H.

AND s

Operation: $A \leftarrow A \wedge s$

Format:

<u>Opcode</u>	<u>Operands</u>
AND	s

The s operand is any of r,n,(HL),(IX+d) or (IY+d), as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:

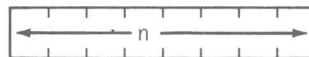
AND r



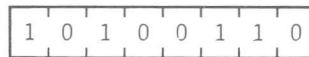
AND n



E6



AND (HL)



A6

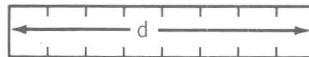
AND (IX+d)



DD



A6



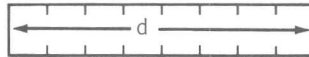
AND (IY+d)



FD



A6



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

<u>Register</u>	<u>r</u>
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

A logical AND operation, bit by bit, is performed between the byte specified by the s operand and the byte contained in the Accumulator; the result is stored in the Accumulator.

<u>INSTRUCTION</u>	<u>M CYCLES</u>	<u>T STATES</u>	<u>4 MHZ E.T.</u>
AND r	1	4	1.00
AND n	2	7(4,3)	1.75
AND (HL)	2	7(4,3)	1.75
AND (IX+d)	5	19(4,4,3,5,3)	4.75
AND (IX+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Set
P/V: Set if parity even;
reset otherwise
N: Reset
C: Reset

Example:

If the B register contains 7BH (01111011) and the Accumulator contains C3H (11000011) after the execution of

AND B

the Accumulator will contain 43H (01000011).

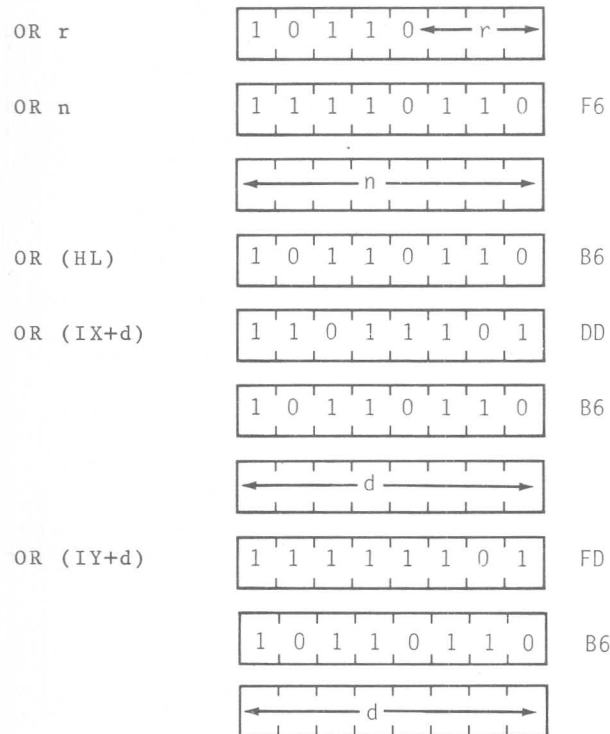
OR s

Operation: $A \leftarrow A \vee s$

Format:

<u>Opcode</u>	<u>Operands</u>
OR	s

The s operand is any of r,n,(HL),(IX+d) or (IY+d), as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

<u>Register</u>	<u>r</u>
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

A logical OR operation, bit by bit, is performed between the byte specified by the s operand and the byte contained in the Accumulator; the result is stored in the Accumulator.

<u>INSTRUCTION</u>	<u>M CYCLES</u>	<u>T STATES</u>	<u>4 MHZ E.T.</u>
OR r	1	4	1.00
OR n	2	7(4,3)	1.75
OR (HL)	2	7(4,3)	1.75
OR (IX+d)	5	19(4,4,3,5,3)	4.75
OR (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Set
P/V: Set if parity even;
reset otherwise
N: Reset
C: Reset

Example:

If the H register contains 48H (010001000) and the Accumulator contains 12H (00010010) after the execution of

OR H

the Accumulator will contain 5AH (01011010).

XOR s

Operation: $A \leftarrow A \oplus s$

Format:

<u>Opcode</u>	<u>Operands</u>
XOR	s

The s operand is any of r,n, (HL), (IX+d) or (IY+d), as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:

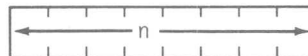
XOR r



XOR n



EE

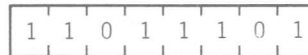


XOR (HL)



AE

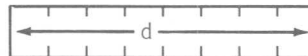
XOR (IX+d)



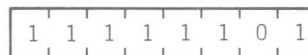
DD



AE



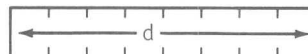
XOR (IY+d)



FD



AE



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

<u>Register</u>	<u>r</u>
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

A logical exclusive-OR operation, bit by bit, is performed between the byte specified by the s operand and the byte contained in the Accumulator; the result is stored in the Accumulator.

<u>INSTRUCTION</u>	<u>M CYCLES</u>	<u>T STATES</u>	<u>4 MHZ E.T.</u>
XOR r	1	4	1.00
XOR n	2	7(4,3)	1.75
XOR (HL)	2	7(4,3)	1.75
XOR (IX+d)	5	19(4,4,3,5,3)	4.75
XOR (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Set
P/V: Set if parity even;
reset otherwise
N: Reset
C: Reset

Example:

If the Accumulator contains 96H (10010110), after the execution of

XOR 5DH (Note: 5DH = 01011101)

the Accumulator will contain CBH (11001011).

CP s

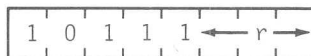
Operation: A - sFormat:OpcodeOperands

CP

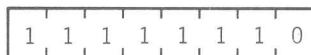
s

The s operand is any of r,n,(HL),(IX+d) or (IY+d), as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:

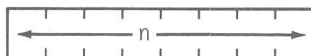
CP r



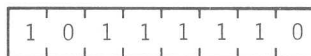
CP n



FE

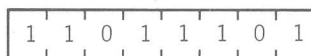


CP (HL)

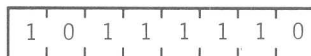


BE

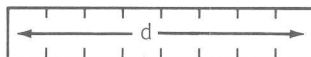
CP (IX+d)



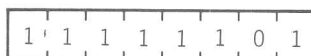
DD



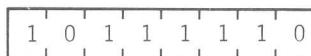
BE



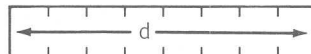
CP (IY+d)



FD



BE



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

<u>Register</u>	<u>r</u>
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The contents of the s operand are compared with the contents of the Accumulator. If there is a true compare, a flag is set.

<u>INSTRUCTION</u>	<u>M CYCLES</u>	<u>T STATES</u>	<u>4 MHZ E.T.</u>
CP r	1	4	1.00
CP n	2	7(4,3)	1.75
CP (HL)	2	7(4,3)	1.75
CP (IX+d)	5	19(4,4,3,5,3)	4.75
CP (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Set if there is a borrow
and reset otherwise.
P/V: Set if overflow;
reset otherwise
N: Set
C: Set if there is a borrow
and reset otherwise.

Example:

If the Accumulator contains 63H, the HL register pair contains 6000H and memory location 6000H contains 60H, the instruction

CP (HL)

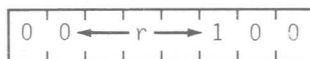
will result in the P/V flag in the F register being reset.

INC r

Operation: $r \leftarrow r + 1$

Format:

<u>Opcode</u>	<u>Operands</u>
INC	r



Description:

Register r is incremented. r identifies any of the registers A,B, C,D,E,H or L, assembled as follows in the object code.

<u>Register</u>	<u>r</u>
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if r was 7FH before operation; reset otherwise
N:	Reset
C:	Not affected

Example:

If the contents of register D are 28H, after the execution of

INC D

the contents of register D will be 29H.

INC (HL)

Operation: (HL) $\leftarrow$ (HL)+1

Format:

<u>Opcode</u>	<u>Operands</u>
INC	(HL)

00110100	34
---	----

Description:

The byte contained in the address specified by the contents of the HL register pair is incremented.

M CYCLES: 3 T STATES: 11(4,4,3) 4 MHZ E.T.: 2.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if (HL) was 7FH before operation; reset otherwise
N:	Reset
C:	Not Affected

Example:

If the contents of the HL register pair are 3434H, and the contents of address 3434H are 82H, after the execution of

INC (HL)

memory location 3434H will contain 83H.

INC (IX+d)

Format:

Description:

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected:

```

S:   Set if result is negative;
      reset otherwise
Z:   Set if result is zero;
      reset otherwise
H:   Set if carry from
      Bit 3; reset otherwise
P/V: Set if (IX+d) was 7FH before
      operation; reset otherwise
N:   Reset
C:   Not affected

```

Example:

If the contents of the Index Register, pair IX are 2020H,
and the memory location 2030H contains byte 34H, after
the execution of

INC (IX+10H)

the contents of memory location 2030H will be 35H.

INC (IY+d)

Operation: $(IY+d) \leftarrow (IY+d)+1$

Format:

<u>Opcode</u>	<u>Operands</u>
INC	(IY+d)

1 1 1 1 1 1 0 1	FD
-----------------	----

0 0 1 1 0 1 0 0	34
-----------------	----

← d →

Description:

The contents of the Index Register IY (register pair IY) are added to a two's complement displacement integer d to point to an address in memory. The contents of this address are then incremented.

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if (IY+d) was 7FH before operation; reset otherwise
N:	Reset
C:	Not Affected

Example:

If the contents of the Index Register pair IY are 2020H,
and the memory location 2030H contain byte 34H, after
the execution of

INC (IY+10H)

the contents of memory location 2030H will be 35H.

DEC m

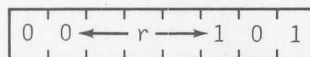
Operation: $m \leftarrow m-1$

Format:

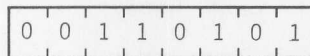
<u>Opcode</u>	<u>Operands</u>
DEC	m

The m operand is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous INC instructions. These various possible opcode-operand combinations are assembled as follows in the object code:

DEC r

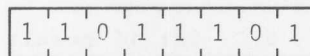


DEC (HL)

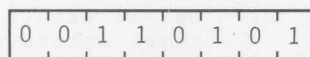


35

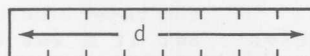
DEC (IX+d)



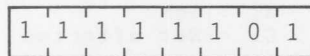
DD



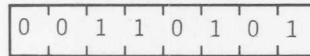
35



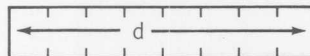
DEC (IY+d)



FD



35



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

<u>Register</u>	<u>r</u>
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The byte specified by the m operand is decremented.

<u>INSTRUCTION</u>	<u>M CYCLES</u>	<u>T STATES</u>	<u>4 MHZ E.T.</u>
DEC r	1	4	1.00
DEC (HL)	3	11(4,4,3)	2.75
DEC (IX+d)	6	23(4,4,3,5,4,3)	5.75
DEC (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Set if there is a borrow
and reset otherwise.
P/V: Set if m was 80H before
operation; reset otherwise
N: Set
C: Not affected

Example:

If the D register contains byte 2AH, after the execution of

DEC D

register D will contain 29H.

2.5 - GENERAL PURPOSE ARITHMETIC AND CPU CONTROL GROUPS -

NAME	VALUE	UNIT	NAME	VALUE	UNIT
0000	0.0	0	0001	0.0	0
0002	0.0	0	0003	0.0	0
0004	0.0	0	0005	0.0	0
0006	0.0	0	0007	0.0	0
0008	0.0	0	0009	0.0	0
0010	0.0	0	0011	0.0	0
0012	0.0	0	0013	0.0	0
0014	0.0	0	0015	0.0	0
0016	0.0	0	0017	0.0	0
0018	0.0	0	0019	0.0	0
0020	0.0	0	0021	0.0	0
0022	0.0	0	0023	0.0	0
0024	0.0	0	0025	0.0	0
0026	0.0	0	0027	0.0	0
0028	0.0	0	0029	0.0	0
0030	0.0	0	0031	0.0	0
0032	0.0	0	0033	0.0	0
0034	0.0	0	0035	0.0	0
0036	0.0	0	0037	0.0	0
0038	0.0	0	0039	0.0	0
0040	0.0	0	0041	0.0	0
0042	0.0	0	0043	0.0	0
0044	0.0	0	0045	0.0	0
0046	0.0	0	0047	0.0	0
0048	0.0	0	0049	0.0	0
0050	0.0	0	0051	0.0	0
0052	0.0	0	0053	0.0	0
0054	0.0	0	0055	0.0	0
0056	0.0	0	0057	0.0	0
0058	0.0	0	0059	0.0	0
0060	0.0	0	0061	0.0	0
0062	0.0	0	0063	0.0	0
0064	0.0	0	0065	0.0	0
0066	0.0	0	0067	0.0	0
0068	0.0	0	0069	0.0	0
0070	0.0	0	0071	0.0	0
0072	0.0	0	0073	0.0	0
0074	0.0	0	0075	0.0	0
0076	0.0	0	0077	0.0	0
0078	0.0	0	0079	0.0	0
0080	0.0	0	0081	0.0	0
0082	0.0	0	0083	0.0	0
0084	0.0	0	0085	0.0	0
0086	0.0	0	0087	0.0	0
0088	0.0	0	0089	0.0	0
0090	0.0	0	0091	0.0	0
0092	0.0	0	0093	0.0	0
0094	0.0	0	0095	0.0	0
0096	0.0	0	0097	0.0	0
0098	0.0	0	0099	0.0	0

DAA

Operation: —Format:Opcode

DAA

0	0	1	0	0	1	1	1
---	---	---	---	---	---	---	---

27

Description:

This instruction conditionally adjusts the Accumulator for BCD addition and subtraction operations. For addition (ADD, ADC, INC) or subtraction (SUB, SBC, DEC, NEG), the following table indicates operation performed:

OPERATION	C BEFORE DAA	HEX VALUE IN UPPER DIGIT (bit 7-4)	H BEFORE DAA	HEX VALUE IN LOWER DIGIT (bit 3-0)	NUMBER ADDED TO BYTE	C AFTER DAA
ADD ADC INC	0	0-9	0	0-9	00	0
	0	0-8	0	A-F	06	0
	0	0-9	1	0-3	06	0
	0	A-F	0	0-9	60	1
	0	9-F	0	A-F	66	1
	0	A-F	1	0-3	66	1
	1	0-2	0	0-9	60	1
	1	0-2	0	A-F	66	1
	1	0-3	1	0-3	66	1
SUB SBC DEC NEG	0	0-9	0	0-9	00	0
	0	0-8	1	6-F	FA	0
	1	7-F	0	0-9	A0	1
	1	6-F	1	6-F	9A	1

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S: Set if most significant bit
of Acc. is 1 after operation;
reset otherwise
Z: Set if Acc. is zero after operation;
reset otherwise
H: See instruction
P/V: Set if Acc. is even parity after
operation; reset otherwise
N: Not affected
C: See instruction

Example:

If an addition operation is performed between 15 (BCD)
and 27 (BCD), simple decimal arithmetic gives this
result:

$$\begin{array}{r} 15 \\ +27 \\ \hline 42 \end{array}$$

But when the binary representations are added in the
Accumulator according to standard binary arithmetic,

$$\begin{array}{r} 0001 \ 0101 \\ +0010 \ 0111 \\ \hline 0011 \ 1100 \quad 3C \end{array}$$

the sum is ambiguous. The DAA instruction adjusts this
result so that the correct BCD representation is
obtained:

$$\begin{array}{r} 0011 \ 1100 \\ +0000 \ 0110 \\ \hline 0100 \ 0010 = 42 \end{array}$$

CPL

Operation: $A \leftarrow \bar{A}$

Format:

Opcode

CPL

0	0	1	0	1	1	1	1	2F
---	---	---	---	---	---	---	---	----

Description:

Contents of the Accumulator (register A) are inverted (1's complement).

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Set
P/V:	Not affected
N:	Set
C:	Not affected

Example:

If the contents of the Accumulator are 1011 0100, after the execution of

CPL

the Accumulator contents will be 0100 1011.

NEG

Operation: $A \leftarrow \neg A$

Format:

Opcode

NEG

1	1	1	0	1	1	0	1
---	---	---	---	---	---	---	---

ED

0	1	0	0	0	1	0	0
---	---	---	---	---	---	---	---

44

Description:

Contents of the Accumulator are negated (two's complement). This is the same as subtracting the contents of the Accumulator from zero. Note that 80H is left unchanged.

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected:

S: Set if result is negative;
 reset otherwise
Z: Set if result is zero;
 reset otherwise
H: Set if there is a borrow
 and reset otherwise.
P/V: Set if Acc. was 80H before
 operation; reset otherwise
N: Set
C: Set if Acc. was not 00H before
 operation; reset otherwise

Example:

If the contents of the Accumulator are

1	0	0	1	1	0	0	0
---	---	---	---	---	---	---	---

after the execution of

NEG

the Accumulator contents will be

0	1	1	0	1	0	0	0
---	---	---	---	---	---	---	---

CCF

Operation: $CY \leftarrow \overline{CY}$

Format:

Opcode

CCF

0	0	1	1	1	1	1	1
---	---	---	---	---	---	---	---

3F

Description:

The C flag in the F register is inverted.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S: Not affected
 Z: Not affected
 H: Previous carry will be copied
 P/V: Not affected
 N: Reset
 C: Set if CY was 0 before
 operation; reset otherwise

2.5.5

SCF

Operation: $CY \leftarrow 1$

Format:

Opcode

SCF

0	0	1	1	0	1	1	1
---	---	---	---	---	---	---	---

 37

Description:

The C flag in the F register is set.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Reset
P/V:	Not affected
N:	Reset
C:	Set

25.6

NOP

Operation: —

Format:

Opcode

NOP

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

 00

Description:

CPU performs no operation during this machine cycle.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: None

2.5.7

HALT

Operation: —

Format:

Opcode

HALT

0	1	1	1	0	1	1	0
---	---	---	---	---	---	---	---

 76

Description:

The HALT instruction suspends CPU operation until a subsequent interrupt or reset is received. While in the halt state, the processor will execute NOP's to maintain memory refresh logic.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: None

DI

Operation: $IFF \leftarrow 0$

Format:

Opcode

DI

1	1	1	1	0	0	1	1
---	---	---	---	---	---	---	---

 F3

Description:

DI disables the maskable interrupt by resetting the interrupt enable flip-flops (IFF1 and IFF2). Note that this instruction disables the maskable interrupt during its execution.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: None

Example:

When the CPU executes the instruction

DI

the maskable interrupt is disabled until it is subsequently re-enabled by an EI instruction. The CPU will not respond to an Interrupt Request (INT) signal.

Operation: $IFF \leftarrow 1$

Format:

Opcode

EI

1	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---

 FB

Description:

EI enables the maskable interrupt by setting the interrupt enable flip-flops (IFF1 and IFF2). Note that this instruction disables the maskable interrupt during its execution.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: None

Example:

When the CPU executes instruction

EI

the maskable interrupt is enabled. The CPU will now respond to an Interrupt Request (INT) signal.

IM 0

Operation: —Format:

<u>Opcode</u>	<u>Operands</u>
IM	0
1 1 1 0 1 1 0 1	ED
0 1 0 0 0 1 1 0	46

Description:

The IM 0 instruction sets interrupt mode 0. In this mode the interrupting device can insert any instruction on the data bus and allow the CPU to execute it.

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected: None

IM 1

Operation: —Format:

<u>Opcode</u>	<u>Operands</u>
IM	1
1 1 1 0 1 1 0 1	ED
0 1 0 1 0 1 1 0	56

Description:

The IM instruction sets interrupt mode 1. In this mode the processor will respond to an interrupt by executing a restart to location 0038H.

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected: None

IM 2

Operation: —Format:

<u>Opcode</u>	<u>Operands</u>
IM	2

1 1 1 0 1 1 0 1	ED
0 1 0 1 1 1 1 0	5E

Description:

The IM 2 instruction sets interrupt mode 2. This mode allows an indirect call to any location in memory. With this mode the CPU forms a 16-bit memory address. The upper eight bits are the contents of the Interrupt Vector Register I and the lower eight bits are supplied by the interrupting device.

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected: None

2.6 - 16 BIT ARITHMETIC GROUP -

2.6.1

ADD HL, SS

Operation: $HL \leftarrow HL + ss$

Format:

<u>Opcode</u>	<u>Operands</u>
ADD	HL, ss

0	0	s	s	1	0	0	1
---	---	---	---	---	---	---	---

Description:

The contents of register pair ss (any of register BC, DE, HL or SP) are added to the contents of register pair HL and the result is stored in HL. Operand ss is specified as follows in the assembled object code

<u>Register</u>	<u>ss</u>
BC	00
DE	01
HL	10
SP	11

M CYCLES: 3 T STATES: 11(4,4,3) 4 MHZ E.T.:

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Set if carry out of Bit 11; reset otherwise
P/V:	Not affected
N:	Reset
C:	Set if carry from Bit 15; reset otherwise

Example:

If register pair HL contains the integer 4242H and register pair DE contains 1111H, after the execution of

ADD HL,DE

the HL register pair will contain 5353H.

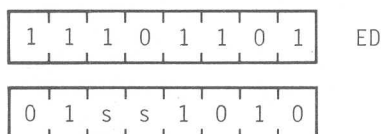
2.6.2

ADC HL, SS

Operation: $HL \leftarrow HL + ss + CY$

Format:

<u>Opcode</u>	<u>Operands</u>
ADC	HL,ss



Description:

The contents of register pair ss (any of register pairs BC,DE,HL or SP) are added with the Carry Flag (C flag in the F register) to the contents of register pair HL, and the result is stored in HL. Operand ss is specified as follows in the assembled object code.

<u>Register</u> <u>Pair</u>	<u>ss</u>
BC	00
DE	01
HL	10
SP	11

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry out of Bit 11; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Reset
C:	Set if carry from Bit 15; reset otherwise

Example:

If the register pair BC contains 2222H, register pair HL contains 5437H and the Carry Flag is set, after the execution of

ADC HL,BC

the contents of HL will be 765AH.

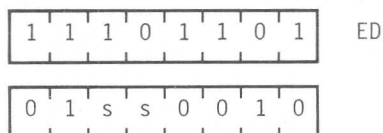
2.6.3

SBC HL, ss

Operation: HL ← HL - ss - CY

Format:

<u>Opcode</u>	<u>Operands</u>
SBC	HL, ss



Description:

The contents of the register pair ss (any of register pairs BC, DE, HL or SP) and the Carry Flag (C flag in the F register) are subtracted from the contents of register pair HL and the result is stored in HL. Operand ss is specified as follows in the assembled object code.

<u>Register</u>	<u>ss</u>
<u>Pair</u>	
BC	00
DE	00
HL	10
SP	11

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if there is a borrow and reset otherwise.
P/V:	Set if overflow; reset otherwise
N:	Set
C:	Set if there is a borrow and reset otherwise.

Example:

If the contents of the HL register pair are 9999H, the contents of register pair DE are 1111H, and the Carry Flag is set, after the execution of

SBC HL,DE

the contents of HL will be 8887H.

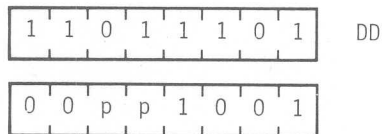
2.6.4

ADD IX, pp

Operation: $IX \leftarrow IX + pp$

Format:

<u>Opcode</u>	<u>Operands</u>
ADD	IX, pp



Description:

The contents of register pair pp (any of register pairs BC, DE, IX or SP) are added to the contents of the Index Register IX, and the results are stored in IX. Operand pp is specified as follows in the assembled object code.

<u>Register</u>	<u>pp</u>
BC	00
DE	01
IX	10
SP	11

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Set if carry out of Bit 11; reset otherwise
P/V:	Not affected
N:	Reset
C:	Set if carry from Bit 15; reset otherwise

Example:

If the contents of Index Register IX are 333H and the contents of register pair BC are 5555H, after the execution of

ADD IX,BC

the contents of IX will be 8888H.

ADD IY, rr

Operation: $IY \leftarrow IY + rr$

Format:

<u>Opcode</u>	<u>Operands</u>
ADD	IY,rr
11111101 FD 00rr1001	

Description:

The contents of register pair rr (any of register pairs BC,DE,IY or SP) are added to the contents of Index Register IY, and the result is stored in IY. Operand rr is specified as follows in the assembled object code.

<u>Register</u> <u>Pair</u>	<u>rr</u>
BC	00
DE	01
IY	10
SP	11

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

S: Not affected
 Z: Not affected
 H: Set if carry out of
 Bit 11; reset otherwise
 P/V: Not affected
 N: Reset
 C: Set if carry from
 Bit 15; reset otherwise

Example:

If the contents of Index Register IY are 333H and the contents of register pair BC are 555H, after the execution of

ADD IY,BC

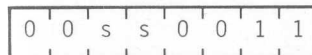
the contents of IY will be 8888H.

INC ss

Operation: $ss \leftarrow ss + 1$

Format:

<u>Opcodes</u>	<u>Operands</u>
INC	ss



Description:

The contents of register pair ss (any of register pairs BC, DE, HL or SP) are incremented. Operand ss is specified as follows in the assembled object code.

<u>Register Pair</u>	<u>ss</u>
BC	00
DE	01
HL	10
SP	11

M CYCLES: 1 T STATES: 6 4 MHZ E.T. 1.50

Condition Bits Affected: None

Example:

If the register pair contains 1000H, after the execution of

INC HL

HL will contain 1001H.

INC IX

Operation: $IX \leftarrow IX + 1$

Format:

<u>Opcode</u>	<u>Operands</u>
INC	IX
1 1 0 1 1 1 0 1	DD
0 0 1 0 0 0 1 1	23

Description:

The contents of the Index Register IX are incremented.

M CYCLES: 2 T STATES: 10(4,6) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the Index Register IX contains the integer 3300H after the execution of

INC IX

the contents of Index Register IX will be 3301H.

INC IY

Operation: $IY \leftarrow IY + 1$

Format:

<u>Opcode</u>	<u>Operands</u>
INC	IY

1	1	1	1	1	1	0	1	FD
0	0	1	0	0	0	1	1	23

Description:

The contents of the Index Register IY are incremented.

M CYCLES: 2 T STATES: 10(4,6) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the contents of the Index Register are 2977H, after the execution of

INC IY

the contents of Index Register IY will be 2978H.

DEC ss

Operation: $ss \leftarrow ss - 1$

Format:

<u>Opcode</u>	<u>Operands</u>
DEC	ss

0	0	s	s	1	0	1	1
---	---	---	---	---	---	---	---

Description:

The contents of register pair ss (any of the register pairs BC,DE,HL or SP) are decremented. Operand ss is specified as follows in the assembled object code.

<u>Pair</u>	<u>ss</u>
BC	00
DE	01
HL	10
SP	11

M CYCLES: 1 T STATES: 6 4 MHZ E.T.: 1.50

Condition Bits Affected: None

Example:

If register pair HL contains 1001H, after the execution of

DEC HL

the contents of HL will be 1000H.

DEC IX

Operation: $IX \leftarrow IX - 1$

Format:

<u>Opcode</u>	<u>Operands</u>
DEC	IX

1 1 0 1 1 1 0 1	DD
0 0 1 0 1 0 1 1	2B

Description:

The contents of Index Register IX are decremented.

M CYCLES: 2 T STATES: 10(4,6) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the contents of Index Register IX are 2006H, after the execution of

DEC IX

the contents of Index Register IX will be 2005H.

DEC IY

Operation: $IY \leftarrow IY - 1$

Format:

<u>Opcode</u>	<u>Operands</u>
DEC	IY

1	1	1	1	1	1	0	1	FD
0	0	1	0	1	0	1	1	2B

Description:

The contents of the Index Register IY are decremented.

M CYCLES: 2 T STATES: 10 (4,6) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the contents of the Index Register IY are 7649H,
after the execution of

DEC IY

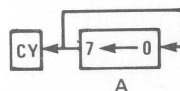
the contents of Index Register IY will be 7648H.

2.7. - ROTATE AND SHIFT GROUP -

2.7.1

RLCA

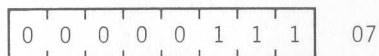
Operation:



Format:

<u>Opcode</u>	<u>Operands</u>
---------------	-----------------

RLCA



Description:

The contents of the Accumulator (register A) are rotated left: the content of bit 0 is moved to bit 1; the previous content of bit 1 is moved to bit 2; this pattern is continued throughout the register. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit 0. (Bit 0 is the least significant bit.)

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Reset
P/V:	Not affected
N:	Reset
C:	Data from Bit 7 of Acc.

Example:

If the contents of the Accumulator are

7 6 5 4 3 2 1 0

1	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

after the execution of

RLCA

the contents of the Accumulator and Carry Flag will be

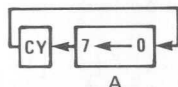
C 7 6 5 4 3 2 1 0

1	0	0	0	1	0	0	0	1
---	---	---	---	---	---	---	---	---

2.7.2

RLA

Operation:



Format:

Opcode

Operands

RLA

0	0	0	1	0	1	1	1
---	---	---	---	---	---	---	---

17

Description:

The contents of the Accumulator (register A) are rotated left: the content of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this pattern is continued throughout the register. The content of bit 7 is copied into the Carry Flag (C flag in register F) and the previous content of the Carry Flag is copied into bit 0. Bit 0 is the least significant bit.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Reset
P/V:	Not affected
N:	Reset
C:	Data from Bit 7 of Acc.

Example:

If the contents of the Accumulator and the Carry Flag are

C 7 6 5 4 3 2 1 0

1	0	1	1	1	0	1	1	0
---	---	---	---	---	---	---	---	---

after the execution of

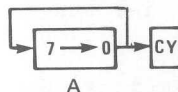
RLA

the contents of the Accumulator and the Carry Flag will be

C 7 6 5 4 3 2 1 0

0	1	1	1	0	1	1	0	1
---	---	---	---	---	---	---	---	---

RRCA

Operation:Format:OpcodeOperands

RRCA

0	0	0	0	1	1	1	1
---	---	---	---	---	---	---	---

 OF
Description:

The contents of the Accumulator (register A) is rotated right: the content of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the register. The content of bit 0 is copied into bit 7 and also into the Carry Flag (C flag in register F.) Bit 0 is the least significant bit.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S: Not affected
 Z: Not affected
 H: Reset
 P/V: Not affected
 N: Reset
 C: Data from Bit 0 of Acc.

Example:

If the contents of the Accumulator are

7 6 5 4 3 2 1 0

0	0	0	1	0	0	0	1
---	---	---	---	---	---	---	---

After the execution of

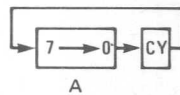
RRCA

the contents of the Accumulator and the Carry Flag will
be

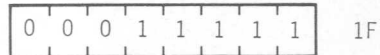
7 6 5 4 3 2 1 0 C

1	0	0	0	1	0	0	0	1
---	---	---	---	---	---	---	---	---

RRA

Operation:Format:OpcodeOperands

RRA

Description:

The contents of the Accumulator (register A) are rotated right: the content of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the register. The content of bit 0 is copied into the Carry Flag (C flag in register F) and the previous content of the Carry Flag is copied into bit 7. Bit 0 is the least significant bit.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S: Not affected
 Z: Not affected
 H: Reset
 P/V: Not affected
 N: Reset
 C: Data from Bit 0 of Acc.

Example:

If the contents of the Accumulator and the Carry Flag are

7 6 5 4 3 2 1 0 C

1	1	1	0	0	0	0	1	0
---	---	---	---	---	---	---	---	---

after the execution of

RRA

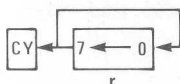
the contents of the Accumulator and the Carry Flag will be

7 6 5 4 3 2 1 0 C

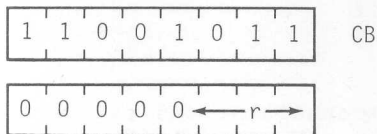
0	1	1	1	0	0	0	0	1
---	---	---	---	---	---	---	---	---

2.7.5

RLC r

Operation:Format:

<u>Opcode</u>	<u>Operands</u>
RLC	r

Description:

The eight-bit contents of register r are rotated left: the content of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this pattern is continued throughout the register. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit 0. Operand r is specified as follows in the assembled object code:

<u>Register</u>	<u>r</u>
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Note: Bit 0 is the least significant bit.

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Reset
P/V: Set if parity even;
reset otherwise
N: Reset
C: Data from Bit 7 of
source register

Example:

If the contents of register r are

7 6 5 4 3 2 1 0

1	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

after the execution of

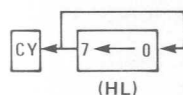
RLC r

the contents of register r and the Carry Flag will be

C 7 6 5 4 3 2 1 0

1	0	0	0	1	0	0	0	1
---	---	---	---	---	---	---	---	---

RLC (HL)

Operation:Format:

<u>Opcode</u>	<u>Operands</u>
RLC	(HL)

1	1	0	0	1	0	1	1	CB
0	0	0	0	0	1	1	0	06

Description:

The contents of the memory address specified by the contents of register pair HL are rotated left: the content of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this pattern is continued throughout the byte. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit 0. Bit 0 is the least significant bit.

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

S: Set if result is negative;
 reset otherwise
 Z: Set if result is zero;
 reset otherwise
 H: Reset
 P/V: Set if parity even;
 reset otherwise
 N: Reset
 C: Data from Bit 7 of
 source register

Example:

If the contents of the HL register pair are 2828H, and the contents of memory location 2828H are

7 6 5 4 3 2 1 0

1	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

after the execution of

RLC (HL)

the contents of memory location 2828H and the Carry Flag will be

C 7 6 5 4 3 2 1 0

1	0	0	0	1	0	0	1
---	---	---	---	---	---	---	---

2.7.7

RLC (IX+d)

Format:

<u>Opcode</u>	<u>Operands</u>
RLC	(IX+d)
	DD
	CB
	06

Description:

The contents of the memory address specified by the sum of the contents of the Index Register IX and a two's complement displacement integer d, are rotated left: the contents of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this pattern is continued throughout the byte. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit 0. Bit 0 is the least significant bit.

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Reset
P/V: Set if parity even;
reset otherwise
N: Reset
C: Data from Bit 7 of
source register

Example:

If the contents of the Index Register IX are 1000H, and the contents of memory location 1022H are

7 6 5 4 3 2 1 0

1	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

after the execution of

RLC (IX+2H)

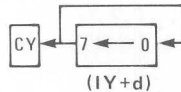
the contents of memory location 1002H and the Carry Flag will be

C 7 6 5 4 3 2 1 0

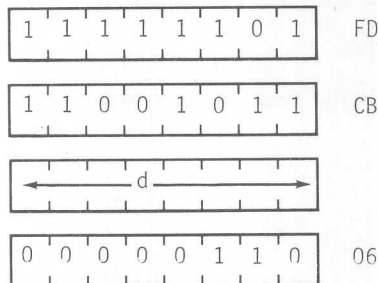
1	0	0	0	1	0	0	0	1
---	---	---	---	---	---	---	---	---

2.7.8

RLC (IY+d)

Operation:Format:

Opcode	Operands
RLC	(IY+d)

Description:

The contents of the memory address specified by the sum of the contents of the Index Register IY and a two's complement displacement integer d are rotated left: the content of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this process is continued throughout the byte. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit 0. Bit 0 is the least significant bit.

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity even; reset otherwise
N:	Reset
C:	Data from Bit 7 of source register

Example:

If the contents of the Index Register IY are 1000H, and the contents of memory location 1002H are

7 6 5 4 3 2 1 0

1	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

after the execution of

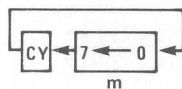
RLC (IY+2H)

the contents of memory location 1002H and the Carry Flag will be

C 7 6 5 4 3 2 1 0

1	0	0	0	1	0	0	0	1
---	---	---	---	---	---	---	---	---

RL m

Operation:Format:Opcode

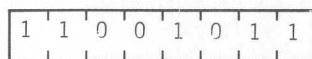
RL

Operands

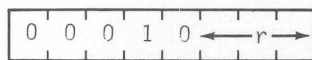
m

The m operand is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

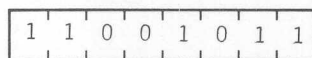
RL r



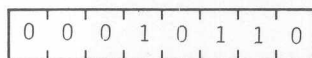
CB



RL (HL)

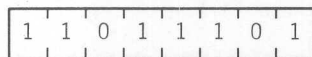


CB

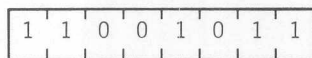


16

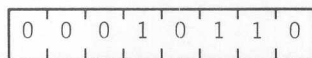
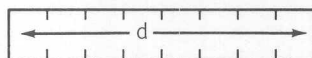
RL (IX+d)



DD

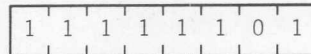


CB

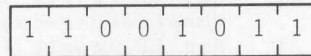


16

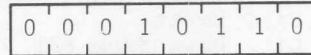
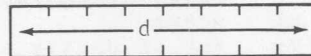
RL (IY+d)



FD



CB



16

*r identifies registers B,C,D,E,H,L or A specified as follows in the assembled object code above:

Register	r
B	000
C	001
D	010
E	011
H	011
L	101
A	111

Description:

The contents of the m operand are rotated left: the content of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this pattern is continued throughout the byte. The content of bit 7 is copied into the Carry Flag (C flag in register F) and the previous content of the Carry Flag is copied into bit 0 (Bit 0 is the least significant bit.)

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
RL r	2	8(4,4)	2.00
RL (HL)	4	15(4,4,4,3)	3.75
RL (IX+d)	6	23(4,4,3,5,4,3)	5.75
RL (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Reset
P/V: Set if parity even;
reset otherwise
N: Reset
C: Data from Bit 7 of
source register

Example:

If the contents of register D and the Carry Flag are

C 7 6 5 4 3 2 1 0

0	1	0	0	0	1	1	1	1
---	---	---	---	---	---	---	---	---

after the execution of

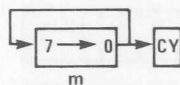
RL D

the contents of register D and the Carry Flag will be

C 7 6 5 4 3 2 1 0

1	0	0	0	1	1	1	1	0
---	---	---	---	---	---	---	---	---

RRC m

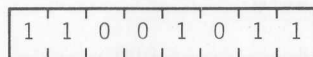
Operation:Format:OpcodeOperands

RRC

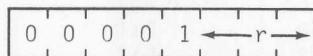
m

The m operand is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

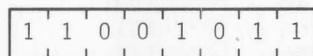
RRC r



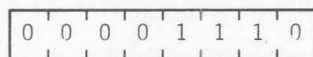
CB



RRC (HL)

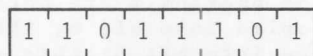


CB

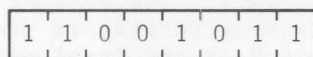


OE

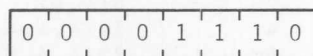
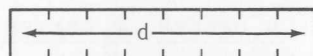
RRC (IX+d)



DD

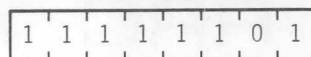


CB

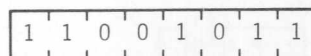


OE

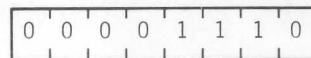
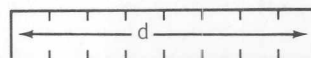
RRC (IY+d)



FD



CB



OE

*r identifies registers B,C,D,E,H,L or A specified as follows in the assembled object code above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The contents of operand m are rotated right: the content of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the byte. The content of bit 0 is copied into the Carry Flag (C flag in the F register) and also into bit 7. Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
RRC r	2	8(4,4)	2.00
RRC (HL)	4	15(4,4,4,3)	3.75
RRC (IX+d)	6	23(4,4,3,5,4,3)	5.75
RRC (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Reset
P/V: Set if parity even;
reset otherwise
N: Reset
C: Data from Bit 0 of
source register

Example:

If the contents of register A are

7 6 5 4 3 2 1 0

0	0	1	1	0	0	0	1
---	---	---	---	---	---	---	---

after the execution of

RRC A

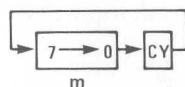
the contents of register A and the Carry Flag will be

7 6 5 4 3 2 1 0 C

1	0	0	1	1	0	0	0	1
---	---	---	---	---	---	---	---	---

2.7.11

RR m

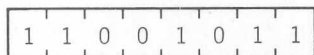
Operation:Format:OpcodeOperand

RR

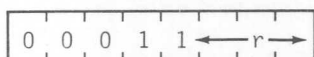
m

The m operand is any of r, (HL), (IX+d), or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

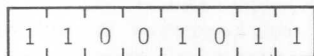
RR r



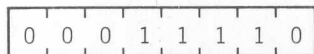
CB



RR (HL)

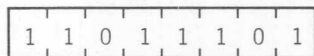


CB

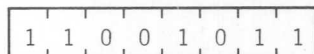


1E

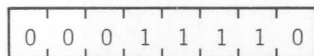
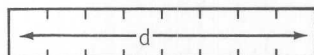
RR (IX+d)



DD

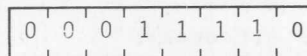


CB

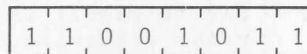


1E

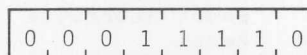
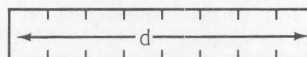
RR (IY+d)



1E



CB



1E

*r identifies registers B,C,D,E,H,L or A specified as follows in the assembled object code above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The contents of operand m are rotated right: the contents of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the byte. The content of bit 0 is copied into the Carry Flag (C flag in register F) and the previous content of the Carry Flag is copied into bit 7. Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
RR r	2	8(4,4)	2.00
RR (HL)	4	15(4,4,4,3)	3.75
RR (IX+d)	6	23(4,4,3,5,4,3)	5.75
RR (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Reset
P/V: Set if parity is even;
reset otherwise
N: Reset
C: Data from Bit 0 of
source register

Example:

If the contents of the HL register pair are 4343H, and the contents of memory location 4343H and the Carry Flag are

7	6	5	4	3	2	1	0	C
1	1	0	1	1	1	0	1	0

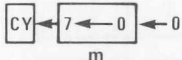
after the execution of

RR (HL)

the contents of location 4343H and the Carry Flag will be

7	6	5	4	3	2	1	0	C
0	1	1	0	1	1	1	0	1

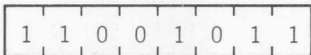
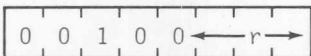
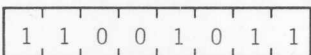
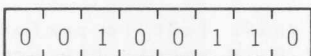
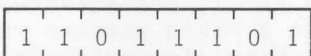
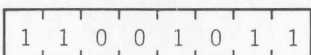
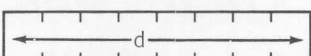
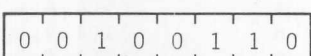
SLA m

Operation: 

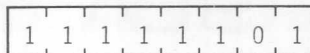
Format:

<u>Opcode</u>	<u>Operands</u>
SLA	m

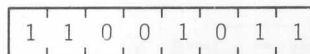
The m operand is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

SLA r		CB
		
SLA (HL)		CB
		26
SLA (IX+d)		DD
		CB
		
		26

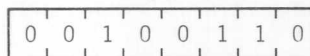
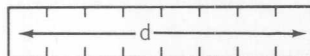
SLA (IY+d)



FD



CB



26

*r identifies registers B,C,D,E,H,L or A specified as follows in the assembled object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

An arithmetic shift left is performed on the contents of operand m: bit 0 is reset, the previous content of bit 0 is copied into bit 1, the previous content of bit 1 is copied into bit 2; this pattern is continued throughout; the content of bit 7 is copied into the Carry Flag (C flag in register F). Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
SLA r	2	8(4,4)	2.00
SLA (HL)	4	15(4,4,4,3)	3.75
SLA (IX+d)	6	23(4,4,3,5,4,3)	5.75
SLA (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Reset
P/V: Set if parity is even;
reset otherwise
N: Reset
C: Data from Bit 7

Example:

If the contents of register L are

7 6 5 4 3 2 1 0

1	0	1	1	0	0	0	1
---	---	---	---	---	---	---	---

after the execution of

SLA L

the contents of register L and the Carry Flag will be

C 7 6 5 4 3 2 1 0

1	0	1	1	0	0	0	1	0
---	---	---	---	---	---	---	---	---

SRA m

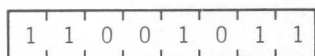


Format:

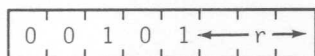
Opcode	Operands
SRA	m

The m operand is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

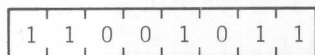
SRA r



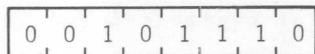
CB



SRA(HL)

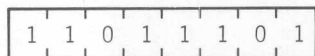


CB

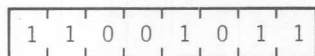


2E

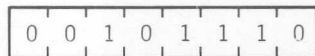
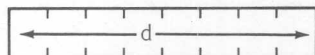
SRA(IX+d)



DD

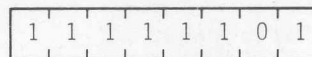


CB

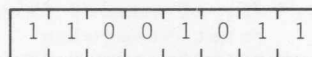


2E

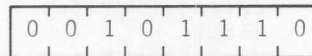
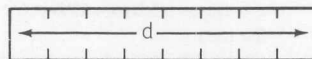
SRA(IY+d)



FD



CB



2E

*r means registers B,C,D,E,H,L or A specified as follows in the assembled object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

An arithmetic shift right is performed on the contents of operand m: the content of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the byte. The content of bit 0 is copied into the Carry Flag (C flag in register F), and the previous content of bit 7 is unchanged. Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
SRA r	2	8(4,4)	2.00
SRA (HL)	4	15(4,4,4,3)	3.75
SRA (IX+d)	6	23(4,4,3,5,4,3)	5.75
SRA (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Reset
P/V: Set if parity is even;
reset otherwise
N: Reset
C: Data from Bit 0 of
source register

Example:

If the contents of the Index Register IX are 1000H, and
the contents of memory location 1003H are

7 6 5 4 3 2 1 0

1	0	1	1	1	0	0	0
---	---	---	---	---	---	---	---

after the execution of

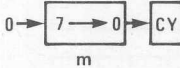
SRA (IX+3H)

the contents of memory location 1003H and the Carry Flag
will be

7 6 5 4 3 2 1 0 C

1	1	0	1	1	1	0	0	0
---	---	---	---	---	---	---	---	---

SRL m

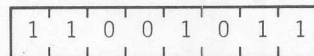
Operation: 

Format:

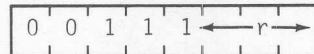
<u>Opcode</u>	<u>Operands</u>
SRL	m

The operand m is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

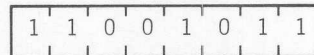
SRL r



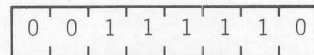
CB



SRL (HL)

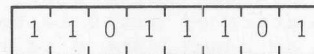


CB

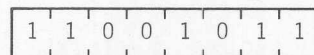


3E

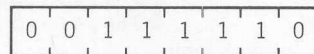
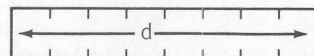
SRL (IX+d)



DD

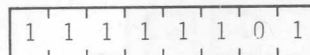


CB

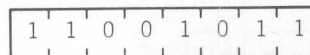


3E

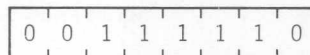
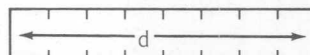
SRL (IY+d)



FD



CB



3E

*r identifies registers B,C,D,E,H,L or A specified as follows in the assembled object code fields above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The contents of operand m are shifted right: the content of bit 7 is copied into bit 6; the content of bit 6 is copied into bit 5; this pattern is continued throughout the byte. The content of bit 0 is copied into the Carry Flag, and bit 7 is reset. Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
SRL r	2	8(4,4)	2.00
SRL (HL)	4	15(4,4,4,3)	3.75
SRL (IX+d)	6	23(4,4,3,5,4,3)	5.75
SRL (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if result is zero;
reset otherwise
H: Reset
P/V: Set if parity is even;
reset otherwise
N: Reset
C: Data from Bit 0 of
source register

Example:

If the contents of register B are

7 6 5 4 3 2 1 0

1	0	0	0	1	1	1	1
---	---	---	---	---	---	---	---

after the execution of

SRL B

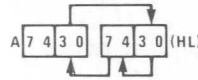
the contents of register B and the Carry Flag will be

7 6 5 4 3 2 1 0 c

0	1	0	0	0	1	1	1	1
---	---	---	---	---	---	---	---	---

RLD

Operation:



Format:

Opcode

Operands

RLD

1	1	1	0	1	1	0	1
---	---	---	---	---	---	---	---

ED

0	1	1	0	1	1	1	1
---	---	---	---	---	---	---	---

6F

Description:

The contents of the low order four bits (bits 3,2,1 and 0) of the memory location (HL) are copied into the high order four bits (7,6,5 and 4) of that same memory location; the previous contents of those high order four bits are copied into the low order four bits of the Accumulator (register A); and the previous contents of the low order four bits of the Accumulator are copied into the low order four bits of memory location (HL). The contents of the high order bits of the Accumulator are unaffected. Note: (HL) means the memory location specified by the contents of the HL register pair.

M CYCLES: 5 T STATES: 18(4,4,3,4,3) 4 MHZ E.T.: 4.50

Condition Bits Affected:

S: Set if Acc. is negative after operation; reset otherwise
 Z: Set if Acc. is zero after operation; reset otherwise
 H: Reset
 P/V: Set if parity of Acc. is even after operation; reset otherwise
 N: Reset
 C: Not affected

Example:

If the contents of the HL register pair are 5000H, and the contents of the Accumulator and memory location 5000H are

7 6 5 4 3 2 1 0

0	1	1	1	1	0	1	0
---	---	---	---	---	---	---	---

Accumulator

7 6 5 4 3 2 1 0

0	0	1	1	0	0	0	1
---	---	---	---	---	---	---	---

(5000H)

after the execution of

RLD

the contents of the Accumulator and memory location 5000H will be

7 6 5 4 3 2 1 0

0	1	1	1	0	0	1	1
---	---	---	---	---	---	---	---

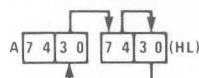
Accumulator

7 6 5 4 3 2 1 0

0	0	0	1	1	0	1	0
---	---	---	---	---	---	---	---

(5000H)

RRD

Operation:Format:

<u>Opcode</u>	<u>Operands</u>
---------------	-----------------

RRD

1	1	1	0	1	1	0	1	ED
0	1	1	0	0	1	1	1	67

Description:

The contents of the low order four bits (bits 3,2,1 and 0) of memory location (HL) are copied into the low order four bits of the Accumulator (register A); the previous contents of the low order four bits of the Accumulator are copied into the high order four bits (7,6,5 and 4) of location (HL); and the previous contents of the high order four bits of (HL) are copied into the low order four bits of (HL). The contents of the high order bits of the Accumulator are unaffected. Note: (HL) means the memory location specified by the contents of the HL register pair.

M CYCLES: 5 T STATES: 18(4,4,3,4,3) 4 MHZ E.T.: 4.50

Condition Bits Affected:

S:	Set if Acc. is negative after operation; reset otherwise
Z:	Set if Acc. is zero after operation; reset otherwise
H:	Reset
P/V:	Set if parity of Acc. is even after operation; reset otherwise
N:	Reset
C:	Not affected

Example:

If the contents of the HL register pair are 5000H, and the contents of the Accumulator and memory location 5000H are

7 6 5 4 3 2 1 0

1	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

Accumulator

7 6 5 4 3 2 1 0

0	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---

(5000H)

after the execution of

RDD

the contents of the Accumulator and memory location 5000H will be

7 6 5 4 3 2 1 0

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

Accumulator

7 6 5 4 3 2 1 0

0	1	0	0	0	0	1	0
---	---	---	---	---	---	---	---

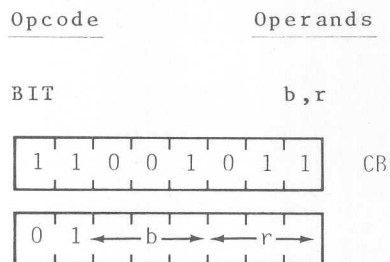
(5000H)

2.8. - BIT SET, RESET AND TEST GROUP -

BIT b, r

Operation: $Z \leftarrow \bar{r}_b$

Format:



Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the indicated register. Operands b and r are specified as follows in the assembled object code:

<u>Bit Tested</u>	<u>b</u>	<u>Register</u>	<u>r</u>
0	000	B	000
1	001	C	001
2	010	D	010
3	011	E	011
4	100	H	100
5	101	L	101
6	110	A	111
7	111		

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected:

S: Unknown
Z: Set if specified Bit is
 0; reset otherwise
H: Set
P/V: Unknown
N: Reset
C: Not affected

Example:

If bit 2 in register B contains 0, after the execution of

BIT 2,B

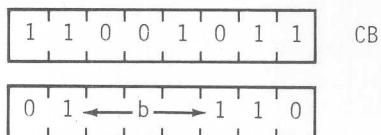
the Z flag in the F register will contain 1, and bit 2 in register B will remain 0. Bit 0 in register B is the least significant bit.

BIT b, (HL)

Operation: $Z \leftarrow \overline{(HL)_b}$

Format:

<u>Opcode</u>	<u>Operands</u>
BIT	b, (HL)



Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the contents of the HL register pair. Operand b is specified as follows in the assembled object code:

<u>Bit Tested</u>	<u>b</u>
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M CYCLES: 3 T STATES: 12(4,4,4) 4 MHZ E.T.: 3.00

Condition Bits Affected:

S:	Unknown
Z:	Set if specified Bit is 0; reset otherwise
H:	Set
P/V:	Unknown
H:	Reset
C:	Not affected

Example:

If the HL register pair contains 4444H, and bit 4 in the memory location 444H contains 1, after the execution of

BIT 4, (HL)

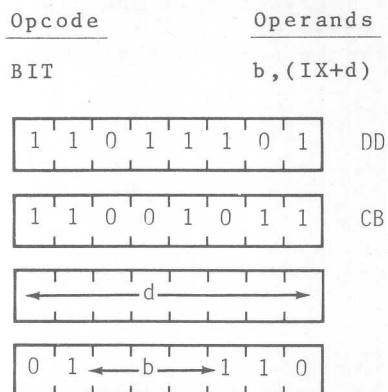
the Z flag in the F register will contain 0, and bit 4 in memory location 4444H will still contain 1. (Bit 0 in memory location 4444H is the least significant bit.)

2.8.3

BIT b, (IX+d)

Operation: $Z \leftarrow \overline{(IX+d)_b}$

Format:



Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the contents of the memory location pointed to by the sum of the contents register pair IX (Index Register IX) and the two's complement displacement integer d. Operand b is specified as follows in the assembled object code.

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M CYCLES: 5 T STATES: 20(4,4,3,5,4) 4 MHZ E.T.: 5.

Condition Bits Affected:

S: Unknown
Z: Set if specified Bit is
0; reset otherwise
H: Set
P/V: Unknown
N: Reset
C: Not affected

Example:

If the contents of Index Register IX are 2000H, and bit 6 in memory location 2004H contains 1, after the execution of

BIT 6, (IX+4H)

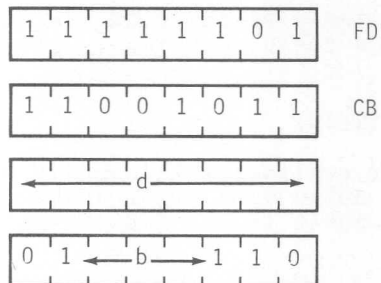
the Z flag in the F register will contain 0, and bit 6 in memory location 2004H will still contain 1. (Bit 0 in memory location 2004H is the least significant bit.)

BIT b, (IY+d)

Operation: $Z \leftarrow \overline{(IY+d)_b}$

Format:

<u>Opcode</u>	<u>Operands</u>
BIT	b, (IY+d)



Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the contents of the memory location pointed to by the sum of the contents of register pair IY (Index Register IY) and the two's complement displacement integer d. Operand b is specified as follows in the assembled object code:

<u>Bit Tested</u>	<u>b</u>
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M CYCLES: 5 T STATES: 20(4,4,3,5,4) 4 MHZ E.T.: 5.00

Condition Bits Affected:

S: Unknown
Z: Set if specified Bit is
0; reset otherwise
H: Set
P/V: Unknown
N: Reset
C: Not affected

Example:

If the contents of Index Register are 2000H, and bit 6 in memory location 2004H contains 1, after the execution of

BIT 6, (IX+4H)

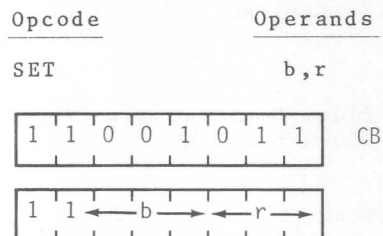
the Z flag in the F register will contain 0, and bit 6 in memory location 2004H will still contain 1. (Bit 0 in memory location 2004H is the least significant bit.)

2.8.5

SET b, r

Operation: $r_b \leftarrow 1$

Format:



Description:

Bit b (any bit, 7 through 0) in register r (any of registers B,C,D,E,H,L or A) is set. Operands b and r are specified as follows in the assembled object code:

<u>Bit</u>	<u>b</u>	<u>Register</u>	<u>r</u>
0	000	B	000
1	001	C	001
2	010	D	010
3	011	E	011
4	100	H	100
5	101	L	101
6	110	A	111
7	111		

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected: None

Example:

After the execution of

SET 4,A

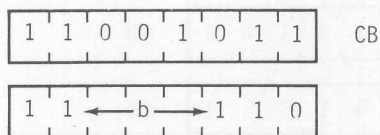
bit 4 in register A will be set. (Bit 0 is the least significant bit.)

SET b, (HL)

Operation: $(HL)_b \leftarrow 1$

Format:

<u>Opcode</u>	<u>Operands</u>
SET	b, (HL)



Description:

Bit b (any bit, 7 through 0) in the memory location addressed by the contents of register pair HL is set. Operand b is specified as follows in the assembled object code:

<u>Bit Tested</u>	<u>b</u>
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected: None

Example:

If the contents of the HL register pair are 3000H, after the execution of

SET 4, (HL)

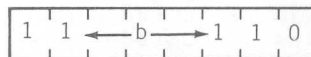
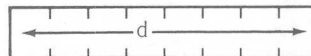
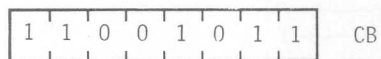
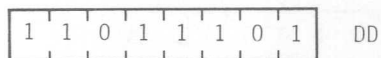
bit 4 in memory location 3000H will be 1. (Bit 0 in memory location 3000H is the least significant bit.)

2.8.7 SET b, (IX+d)

Operation: $(IX+d)_b \leftarrow 1$

Format:

<u>Opcode</u>	<u>Operands</u>
SET	b, (IX+d)



Description:

Bit b (any bit, 7 through 0) in the memory location addressed by the sum of the contents of the IX register pair (Index Register IX) and the two's complement integer d is set. Operand b is specified as follows in the assembled object code:

<u>Bit Tested</u>	<u>b</u>
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected: None

Example:

If the contents of Index Register are 2000H, after the execution of

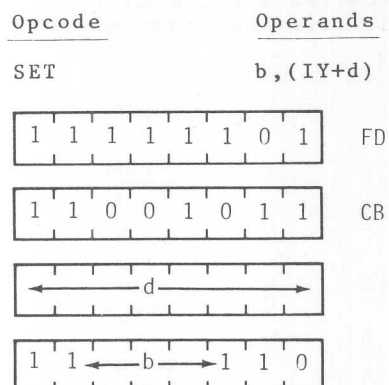
SET 0,(IX+3H)

bit 0 in memory location 2003H will be 1. (Bit 0 in memory location 2003H is the least significant bit.)

2.8.8 SET b, (IY+d)

Operation: $(IY+d)_b \leftarrow 1$

Format:



Description:

Bit b (any bit, 7 through 0) in the memory location addressed by the sum of the contents of the IY register pair (Index Register IY) and the two's complement displacement d is set. Operand b is specified as follows in the assembled object code:

<u>Bit Tested</u>	<u>b</u>
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected: None

Example:

If the contents of Index Register IY are 2000H, after the execution of

SET 0,(IY+3H)

bit 0 in memory location 2003H will be 1. (Bit 0 in memory location 2003H is the least significant bit.)

2.8.9

RES b, m

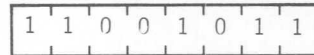
Operation: $s_b \leftarrow 0$

Format:

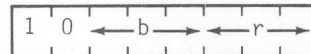
<u>Opcode</u>	<u>Operands</u>
RES	b, m

Operand b is any bit (7 through 0) of the contents of the m operand, (any of r, (HL), (IX+d) or (IY+d)) as defined for the analogous SET instructions. These various possible opcode-operand combinations are assembled as follows in the object code:

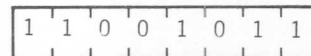
RES b, r



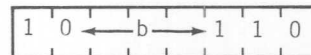
CB



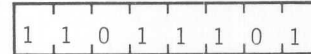
RES b, (HL)



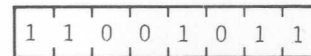
CB



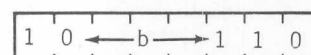
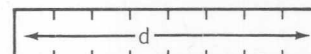
RES b, (IX+d)



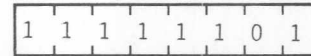
DD



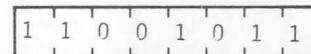
CB



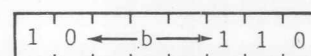
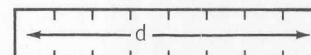
RES b, (IY+d)



FD



CB



<u>Bit Reset</u>	<u>b</u>	<u>Register</u>	<u>r</u>
0	000	B	000
1	001	C	001
2	010	D	010
3	011	E	011
4	100	H	100
5	101	L	101
6	110	A	111
7	111		

Description:

Bit b in operand m is reset.

<u>INSTRUCTION</u>	<u>M CYCLES</u>	<u>T STATES</u>	<u>4 MHZ E.T.</u>
RES r	4	8(4,4)	2.00
RES (HL)	4	15(4,4,4,3)	3.75
RES (IX+d)	6	23(4,4,3,5,4,3)	5.75
RES (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected: None

Example:

After the execution of

RES 6,D

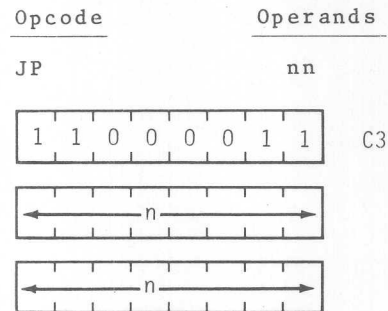
bit 6 in register D will be reset. (Bit 0 in register D is the least significant bit.)

2.9 - JUMP GROUP -

JP nn

Operation: $PC \leftarrow nn$

Format:



Note: The first operand in this assembled object code is the low order byte of a 2-byte address.

Description:

Operand nn is loaded into register pair PC (Program Counter) and points to the address of the next program instruction to be executed.

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

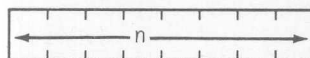
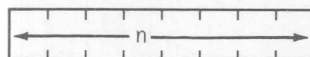
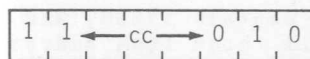
Condition Bits Affected: None

JP cc, nn

Operation: IF cc TRUE, PC $\leftarrow$ nn

Format:

<u>Opcode</u>	<u>Operands</u>
JP	cc, nn



Note: The first n operand in this assembled object code is the low order byte of a 2-byte memory address.

Description:

If condition cc is true, the instruction loads operand nn into register pair PC (Program Counter), and the program continues with the instruction beginning at address nn. If condition cc is false, the Program Counter is incremented as usual, and the program continues with the next sequential instruction. Condition cc is programmed as one of eight status which corresponds to condition bits in the Flag Register (register F). These eight status are defined in the table below which also specifies the corresponding cc bit fields in the assembled object code.

<u>cc</u>	<u>CONDITION</u>	<u>RELEVANT FLAG</u>
000	NZ non zero	Z
001	Z zero	Z
010	NC no carry	C
011	C carry	C
100	PO parity odd	P/V
101	PE parity even	P/V
110	P sign positive	S
111	M sign negative	S

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the Carry Flag (C flag in the F register) is set and the contents of address 1520 are 03H, after the execution of

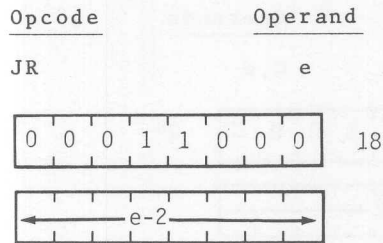
JP C,1520H

the Program Counter will contain 1520H, and on the next machine cycle the CPU will fetch from address 1520H the byte 03H.

JR e

Operation: $PC \leftarrow PC + e$

Format:



Description:

This instruction provides for unconditional branching to other segments of a program. The value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. This jump is measured from the address of the instruction opcode and has a range of -126 to +129 bytes. The assembler calculates the displacement e and automatically adjusts for the twice incremented PC.

M CYCLES: 3 T STATES: 12(4,3,5) 4 MHZ E.T.: 3.00

Condition Bits Affected: None

Example:

To jump forward 5 locations from address 480, the following assembly language statement is used:

JR LABEL

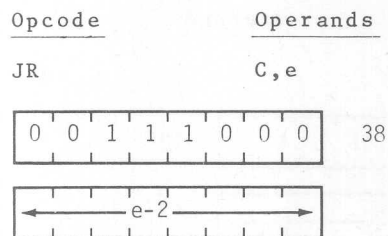
The resulting object code and final PC value is shown below:

Location	Instruction
480 IR LABEL	18
481	03
482	—
483	—
484	—
485 LABEL:	← PC after jump

JR C, e

Operation: If C = 0, continue
 If C = 1, $PC \leftarrow PC + e$

Format:



Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Carry Flag. If the flag is equal to a '1', the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of -126 to +129 bytes. The assembler calculates the displacement e and automatically adjusts for the twice incremented PC.

If the flag is equal to a '0', the next instruction to be executed is taken from the location following this instruction.

If condition is met:

M CYCLES: 3 T STATES: 12(4,3,5) 4 MHZ E.T.: 3.00

If condition is not met:

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

The Carry Flag is set and it is required to jump back 4 locations from 480. The assembly language statement is:

JR C, LABEL

The resulting object code and final PC value is shown below:

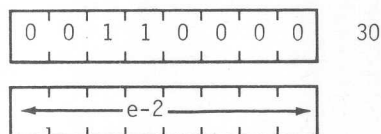
<u>Location</u>	<u>Instruction</u>
47C LABEL:	← PC after jump
47D	—
47E	—
47F	—
480 JR C, LABEL	38
481	FA (2's complement-6)

JR NC, e

Operation: If C = 1, continue
 If C = 0, $PC \leftarrow PC + e$

Format:

<u>Opcode</u>	<u>Operands</u>
JR	NC, e



Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Carry Flag. If the flag is equal to '0', the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of -126 to +129 bytes. The assembler calculates the displacement e and automatically adjusts for the twice incremented PC.

If the flag is equal to a '1', the next instruction to be executed is taken from the location following this instruction.

If the condition is met:

M CYCLES: 3 T STATES: 12(4,3,5) 4 MHZ E.T.: 3.00

If the condition is not met:

M CYCLES: 7 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

The Carry Flag is reset and it is required to repeat the jump instruction. The assembly language statement is:

JR NC, LABEL

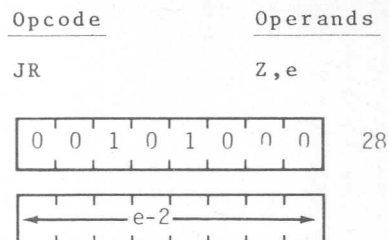
The resulting object code and PC after the jump are shown below:

<u>Location</u>	<u>Instruction</u>
480 LABEL:JR NC, LABEL	30 ← PC after jump
481	00

JR Z, e

Operation: If Z = 0, continue
 If Z = 1, $PC \leftarrow PC + e$

Format:



Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Zero Flag. If the flag is equal to a '1', the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of -126 to +129 bytes. The assembler calculates the displacement e and automatically adjusts for the twice incremented PC.

If the Zero Flag is equal to a '0', the next instruction to be executed is taken from the location following this instruction.

If the condition is met:

M CYCLES: 3 T STATES: 12(4,3,5) 4 MHZ E.T.: 3.00

If the condition is not met:

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

The Zero Flag is set and it is required to jump forward 5 locations from address 300. The following assembly language statement is used:

JR Z, LABEL

The resulting object code and final PC value is shown below:

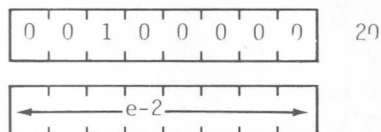
<u>Location</u>	<u>Instruction</u>
300 JR Z, LABEL	28
301	03
302	—
303	—
304	—
305 LABEL:	← PC after jump

JR NZ, e

Operation: If Z = 1, continue
 If Z = 0, $PC \leftarrow PC + e$

Format:

<u>Opcode</u>	<u>Operands</u>
JR	NZ, e



Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Zero Flag. If the flag is equal to a '0', the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of -126 to +129 bytes. The assembler calculates the displacement e and automatically adjusts for the twice incremented PC.

If the Zero Flag is equal to a '1', the next instruction to be executed is taken from the location following this instruction.

If the condition is met:

M CYCLES: 3 T STATES: 12(4,3,5) 4 MHZ E.T.: 3.00

If the condition is not met:

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

The Zero Flag is reset and it is required to jump back 4 locations from 480. The assembly language statement is:

JR NZ, LABEL

The resulting object code and final PC value is shown below:

<u>Location</u>	<u>Instruction</u>
47C LABEL:	← PC after jump
47D	—
47E	—
47F	—
480 JR NZ, LABEL	20
481	FA (2' complement-6)

2.9.8

JP (HL)

Operation: PC $\leftarrow$ HL

Format:

<u>Opcode</u>	<u>Operands</u>
JP	(HL)

1	1	1	0	1	0	0	1	E9
---	---	---	---	---	---	---	---	----

Description:

The Program Counter (register pair PC) is loaded with the contents of the HL register pair. The next instruction is fetched from the location designated by the new contents of the PC.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1000H and the contents of the HL register pair are 4800H, after the execution of

JP (HL)

the contents of the Program Counter will be 4800H.

JP (IX)

Operation: $PC \leftarrow IX$

Format:

<u>Opcode</u>	<u>Operands</u>
JP	(IX)
1 1 0 1 1 1 0 1	DD
1 1 1 0 1 0 0 1	E9

Description:

The Program Counter (register pair PC) is loaded with the contents of the IX Register Pair (Index Register IX). The next instruction is fetched from the location designated by the new contents of the PC.

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1000H, and the contents of the IX Register Pair are 4800H, after the execution of

JP (IX)

the contents of the Program Counter will be 4800H.

JP (IY)

Operation: PC $\leftarrow$ IY

Format:

<u>Opcode</u>	<u>Operands</u>
JP	(IY)

1	1	1	1	1	1	0	1	FD
1	1	1	0	1	0	0	1	E9

Description:

The Program Counter (register pair PC) is loaded with the contents of the IY register pair (Index Register IY). The next instruction is fetched from the location designated by the new contents of the PC.

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1000H and the contents of the IY Register Pair are 4800H, after the execution of

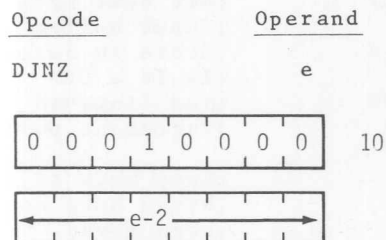
JP (IY)

the contents of the Program Counter will be 4800H.

DJNZ e

Operation: —

Format:



Description:

This instruction is similar to the conditional jump instructions except that a register value is used to determine branching. The B register is decremented and if a non zero value remains, the value of the displacement e is added to the Program Counter (PC). The next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of -126 to +129 bytes. The assembler calculates the displacement e and automatically adjusts for the twice incremented PC.

If the result of decrementing leaves B with a zero value, the next instruction to be executed is taken from the location following this instruction.

If B≠0:

M CYCLES: 3 T STATES: 13(5,3,5) 4 MHZ E.T.: 3.25

If B=0:

M CYCLES: 2 T STATES: 8(5,3) 4 MHZ E.T.: 2.00

Condition Bits Affected: None

Example:

A typical software routine is used to demonstrate the use of the DJNZ instruction. This routine moves a line from an input buffer (INBUF) to an output buffer

(OUTBUF). It moves the bytes until it finds a CR, or until it has moved 80 bytes, whichever occurs first.

```
LD      B,80          ;Set up counter
LD      HL,Inbuf      ;Set up pointers
LD      DE,Outbuf

OOP:    LD      A,(HL)  ;Get next byte from
                        ;input buffer
LD      (DE),A        ;Store in output buffer
CP      00H           ;Is it a CR?
JR      Z,DONE        ;Yes finished
INC     HL            ;Increment pointers
INC     DE
DJNZ    LOOP          ;Loop back if 80
                        ;bytes have not
                        ;been moved

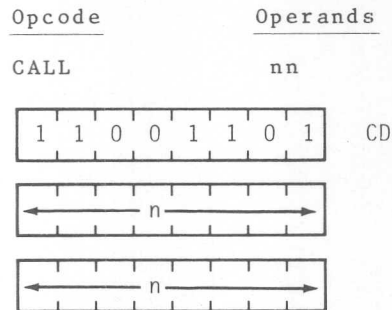
ONE:
```

2.10 CALL AND RETURN GROUP -

CALL nn

Operation: $(SP-1) \leftarrow PC_H$, $(SP-2) \leftarrow PC_L$, $PC \leftarrow nn$

Format:



Note: The first of the two n operands in the assembled object code above is the least significant byte of a two-byte memory address.

Description:

After pushing the current contents of the Program Counter (PC) onto the top of the external memory stack, the operands nn are loaded into PC to point to the address in memory where the first opcode of a subroutine is to be fetched. (At the end of the subroutine, a RETurn instruction can be used to return to the original program flow by popping the top of the stack back into PC.) The push is accomplished by first decrementing the current contents of the Stack Pointer (register pair SP), loading the high-order byte of the PC contents into the memory address now pointed to by the SP; then decrementing SP again, and loading the low-order byte of the PC contents into the top of stack. Note: Because this is a 3-byte instruction, the Program Counter will have been incremented by 3 before the push is executed.

M CYCLES: 5 T STATES: 17(4,3,4,3,3) 4 MHZ E.T.: 4.2

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1A47H, the contents of the Stack Pointer are 3002H, and memory locations have the contents:

Location	Contents
1A47H	CDH
1A48H	35H
1A49H	21H

then if an instruction fetch sequence begins, the three-byte instruction CD3521H will be fetched to the CPU for execution. The mnemonic equivalent of this is

CALL 2135H

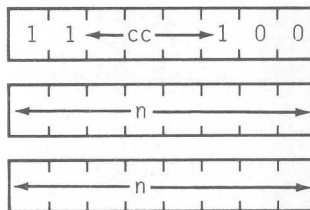
After the execution of this instruction, the contents of memory address 3001H will be 1AH, the contents of address 3000H will be 4AH, the contents of the Stack Pointer will be 3000H, and the contents of the Program Counter will be 2135H, pointing to the address of the first opcode of the subroutine now to be executed.

CALL cc, nn

Operation: IF cc TRUE: $(SP-1) \leftarrow PC_H$
 $(SP-2) \leftarrow PC_L, PC \leftarrow nn$

Format:

<u>Opcode</u>	<u>Operands</u>
CALL	cc, nn



Note: The first of the two n operands in the assembled object code above is the least significant byte of the two-byte memory address.

Description:

If condition cc is true, this instruction pushes the current contents of the Program Counter (PC) onto the top of the external memory stack, then loads the operands nn into PC to point to the address in memory where the first opcode of a subroutine is to be fetched.

(At the end of the subroutine, a RETURN instruction can be used to return to the original program flow by popping the top of the stack back into PC.) If condition cc is false, the Program Counter is incremented as usual, and the program continues with the next sequential instruction. The stack push is accomplished by first decrementing the current contents of the Stack Pointer (SP), loading the high-order byte of the PC contents into the memory address now pointed to by SP; then decrementing SP again, and loading the low-order byte of the PC contents into the top of the stack. Note: Because this is a 3-byte instruction, the Program Counter will have been incremented by 3 before

the push is executed. Condition cc is programmed as one of eight status which corresponds to condition bits in the Flag Register (register F). These eight status are defined in the table below, which also specifies the corresponding cc bit fields in the assembled object code:

cc	Condition	Relevant Flag
000	NZ non zero	Z
001	Z zero	Z
010	NC non carry	C
011	C carry	C
100	PO parity odd	P/V
101	PE parity even	P/V
110	P sign positive	S
111	M sign negative	S

If cc is true:

M CYCLES: 5 T STATES: 17(4,3,4,3,3) 4 MHZ E.T.: 4.25

If cc is false:

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the C Flag in the F register is reset, the contents of the Program Counter are 1A47H, the contents of the Stack Pointer are 3002H, and memory locations have the contents:

Location	Contents
1A47H	D4H
1A48H	35H
1A49H	21H

then if an instruction fetch sequence begins, the three-byte instruction D43521H will be fetched to the CPU for execution. The mnemonic equivalent of this is

CALL NC,2135H

After the execution of this instruction, the contents of memory address 3001H will be 1AH, the contents of address 3000H will be 4AH, the contents of the Stack Pointer will be 3000H, and the contents of the Program Counter will be 2135H, pointing to the address of the first opcode of the subroutine now to be executed.

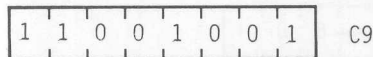
RET

Operation: $PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$

Format:

Opcode

RET



Description:

Control is returned to the original program flow by popping the previous contents of the Program Counter (PC) off the top of the external memory stack, where they were pushed by the CALL instruction. This is accomplished by first loading the low-order byte of the PC with the contents of the memory address pointed to by the Stack Pointer (SP), then incrementing the SP and loading the high-order byte of the PC with the contents of the memory address now pointed to by the SP. (The SP is now incremented a second time.) On the following machine cycle the CPU will fetch the next program opcode from the location in memory now pointed to by the PC.

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 3535H, the contents of the Stack Pointer are 2000H, the contents of memory location 2000H are B5H, and the contents of memory location 2001H are 18H, then after the execution of

RET

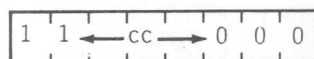
the contents of the Stack Pointer will be 2002H and the contents of the Program Counter will be 18B5H, pointing to the address of the next program opcode to be fetched.

RET cc

Operation: IF cc TRUE: $PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$

Format:

<u>Opcode</u>	<u>Operand</u>
RET	cc



Description:

If condition cc is true, control is returned to the original program flow by popping the previous contents of the Program Counter (PC) off the top of the external memory stack, where they were pushed by the CALL instruction. This is accomplished by first loading the low-order byte of the PC with the contents of the memory address pointed to by the Stack Pointer (SP), then incrementing the SP, and loading the high-order byte of the PC with the contents of the memory address now pointed to by the SP. (The SP is now incremented a second time.) On the following machine cycle the CPU will fetch the next program opcode from the location in memory now pointed to by the PC. If condition cc is false, the PC is simply incremented as usual, and the program continues with the next sequential instruction. Condition cc is programmed as one of eight status which correspond to condition bits in the Flag Register (register F). These eight status are defined in the table below, which also specifies the corresponding cc bit fields in the assembled object code.

<u>cc</u>	<u>Condition</u>	<u>Relevant Flag</u>
000	NZ non zero	Z
001	Z zero	Z
010	NC non carry	C
011	C carry	C
100	PO parity odd	P/V
101	PE parity even	P/V
110	P sign positive	S
111	M sign negative	S

If cc is true:

M CYCLES: 3 T STATES: 11(5,3,3) 4 MHZ E.T.: 2.75

If cc is false:

M CYCLES: 1 T STATES: 5 4 MHZ E.T.: 1.25

Condition Bits Affected: None

Example:

If the S flag in the F register is set, the contents of the Program Counter are 3535H, the contents of the Stack Pointer are 2000H, the contents of memory location 2000H are B5H, and the contents of memory location 2001H are 18H, then after the execution of

RET M

the contents of the Stack Pointer will be 2002H and the contents of the Program Counter will be 18B5H, pointing to the address of the next program opcode to be fetched.

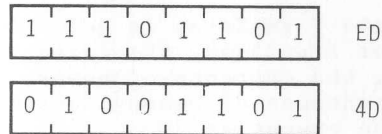
RETI

Operation: Return from interrupt

Format:

Opcode

RETI



Description:

This instruction is used at the end of an interrupt service routine to:

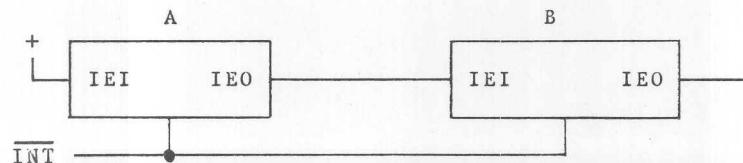
1. Restore the contents of the Program Counter (PC) (analogous to the RET instruction)
2. To signal an I/O device that the interrupt routine has been completed. The RETI instruction facilitates the nesting of interrupts allowing higher priority devices to suspend service of lower priority service routines. The state of IFF2 is copied into IFF1.

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.50

Condition Bits Affected: None

Example:

Given: Two interrupting devices, A and B connected in a daisy chain configuration with A having a higher priority than B.



B generates an interrupt and is acknowledged. (The interrupt enable out, IEO, of B goes low, blocking any lower priority devices from interrupting while B is being serviced). Then A generates an interrupt, suspending service of B. (The IEO of A goes 'low' indicating that a higher priority device is being serviced.) The A routine is completed and a RETI is issued resetting the IEO of A, allowing the B routine to continue. A second RETI is issued on completion of the B routine and the IEO of B is reset (high) allowing lower priority devices interrupt access.

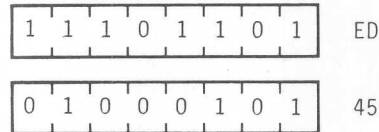
RETN

Operation: Return from non maskable interrupt

Format:

Opcode

RETN



Description:

Used at the end of a service routine for a non maskable interrupt, this instruction executes an unconditional return which functions identical to the RET instruction. That is, the previously stored contents of the Program Counter (PC) are popped off the top of the external memory stack; the low-order byte of PC is loaded with the contents of the memory location pointed to by the Stack Pointer (SP), SP is incremented, the high-order byte of PC is loaded with the contents of the memory location now pointed to by SP, and SP is incremented again. Control is now returned to the original program flow: on the following machine cycle the CPU will fetch the next opcode from the location in memory now pointed to by the PC. Also the state of IFF2 is copied back into IFF1 to the state it had prior to the acceptance of the NMI.

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.50

Condition Bits Affected: None

Example:

If the contents of the Stack Pointer are 1000H and the contents of the Program Counter are 1A45H when a non maskable interrupt (NMI) signal is received, the CPU will ignore the next instruction and will instead restart to memory address 0066H. That is, the current Program Counter contents of 1A45H will be pushed onto the external stack address of 0FFFH and 0FFEh, high

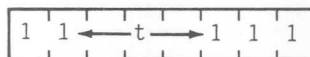
order-byte first, and 0066H will be loaded onto the Program Counter. That address begins an interrupt service routine which ends with RETN instruction. Upon the execution of RETN, the former Program Counter contents are popped off the external memory stack, low-order first, resulting in a Stack Pointer contents again of 1000H. The program flow continues where it left off with an opcode fetch to address 1A45H.

RST p

Operation: $(SP-1) \leftarrow PC_H$, $(SP-2) \leftarrow PC_L$, $PC_H \leftarrow 0$, $PC_L \leftarrow p$

Format:

<u>Opcode</u>	<u>Operand</u>
RST	p



Description:

The current Program Counter (PC) contents are pushed onto the external memory stack, and the page zero memory location given by operand p is loaded into the PC. Program execution then begins with the opcode in the address now pointed to by PC. The push is performed by first decrementing the contents of the Stack Pointer (SP), loading the high-order byte of PC into the memory address now pointed to by SP, decrementing SP again, and loading the low-order byte of PC into the address now pointed to by SP. The ReStart instruction allows for a jump to one of eight addresses as shown in the table below. The operand p is assembled into the object code using the corresponding T state. Note: Since all addresses are in page zero of memory, the high order byte of PC is loaded with 00H. The number selected from the "p" column of the table is loaded into the low-order byte of PC.

<u>p</u>	<u>t</u>
00H	000
08H	001
10H	010
18H	011
20H	100
28H	101
30H	110
38H	111

M CYCLES: 3 T STATES: 11(5,3,3) 4 MHZ E.T.: 2.75

Example:

If the contents of the Program Counter are 15B3H, after the execution of

RST 18H (Object code 1101111)

the PC will contain 0018H, as the address of the next opcode to be fetched.

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES

DEPARTMENT OF PHYSICS

PHYSICS 311
LECTURE 10
MAY 19, 1961

2.11 - INPUT AND OUTPUT GROUP -

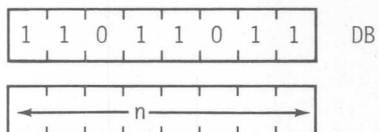
2.11.1

IN A, (n)

Operation: $A \leftarrow (n)$

Format:

<u>Opcode</u>	<u>Operands</u>
IN	A, (n)



Description:

The operand n is placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of the Accumulator also appear on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written into the Accumulator (register A) in the CPU.

M CYCLES: 3 T STATES: 11(4,3,4) 4 MHZ E.T.: 2.75

Condition Bits Affected: None

Example:

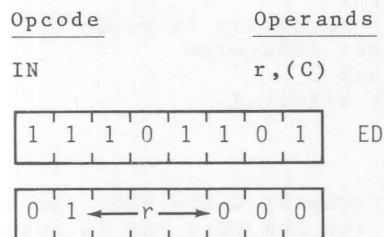
If the contents of the Accumulator are 23H and the byte 7BH is available at the peripheral device mapped to I/O port address 01H, then after the execution of

IN A, (01H)

the Accumulator will contain 7BH.

2.11.2

IN r, (C)

Operation: $r \leftarrow (C)$ Format:Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of Register B are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written into register r in the CPU. Register r identifies any of the CPU registers shown in the following table, which also shows the corresponding 3-bit "r" field for each. The flags will be affected, checking the input data.

<u>Reg.</u>	<u>r</u>
B	000
C	001
D	010
E	011
H	100
L	101
A	111

M CYCLES: 3 T STATES: 12(4,4,4) 4 MHZ E.T.: 3.00

Condition Bits Affected:

S: Set if input data is negative;
reset otherwise
Z: Set if input data is zero;
reset otherwise
H: Reset
P/V: Set if parity is even;
reset otherwise
N: Reset
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 10H, and the byte 7BH is available at the peripheral device mapped to I/O port address 07H, then after the execution of

IN D,(C)

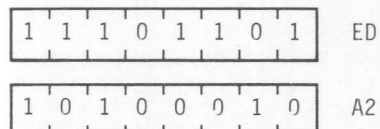
INI

Operation: $(HL) \leftarrow (C)$, $B \leftarrow B-1$, $HL \leftarrow HL + 1$

Format:

Opcode

INI



Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are then placed on the address bus and the input byte is written into the corresponding location of memory. Finally the byte counter is decremented and register pair HL is incremented.

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S: Unknown
 Z: Set if B-1=0;
 reset otherwise
 H: Unknown
 P/V: Unknown
 N: Set
 C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 10H, the contents of the HL register pair are 1000H, and the byte 7BH is available at the peripheral device mapped to I/O port address 07H, then

after the execution of

INI

memory location 1000H will contain 7BH, the HL register pair will contain 1001H, and register B will contain 0FH.

INIR

Operation: $(HL) \leftarrow (C)$, $B \leftarrow B-1$, $HL \leftarrow HL + 1$

Format:

Opcode

INIR

1	1	1	0	1	1	0	1	ED
1	0	1	1	0	0	1	0	B2

Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B is used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written into the corresponding location of memory. Then register pair HL is incremented, the byte counter is decremented. If decrementing causes B to go to zero, the instruction is terminated. If B is not zero, the PC is decremented by two and the instruction repeated. Note that if B is set to zero prior to instruction execution, 256 bytes of data will be input. Also, "Interrupts will be recognized and two refresh cycles will be executed after each data transfer".

If $B \neq 0$:

M CYCLES: 5 T STATES: 21(4,5,3,4,5) 4 MHZ E.T.: 5.25

If $B = 0$:

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S: Unknown
Z: Set
H: Unknown
P/V: Unknown
N: Set
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and the following sequence of bytes are available at the peripheral device mapped to I/O port of address 07H:

51H
A9H
03H

then after the execution of

INIR

the HL register pair will contain 1003H, register B will contain zero, and memory locations will have contents as follows:

Location	Contents
1000H	51H
1001H	A9H
1002H	03H

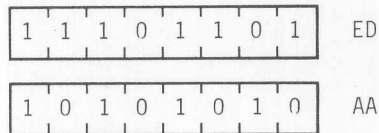
IND

Operation: (HL) $\leftarrow$ (C), B $\leftarrow$ B-1, HL $\leftarrow$ HL-1

Format:

Opcode

IND



Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written into the corresponding location of memory. Finally the byte counter and register pair HL are decremented.

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S: Unknown
 Z: Set if B-1=0;
 reset otherwise
 H: Unknown
 P/V: Unknown
 N: Set
 C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 10H, the contents of the HL register pair are 1000H, and the byte 7BH is available at the

peripheral device mapped to I/O port address 07H, then
after the execution of

IND

memory location 1000H will contain 7BH, the HL register
pair will contain 0FFFH, and register B will contain
0FH.

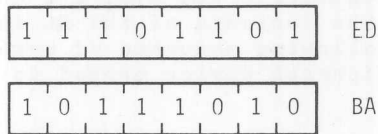
INDR

Operation: $(HL) \leftarrow (C)$, $B \leftarrow B-1$, $HL \leftarrow HL-1$

Format:

Opcode

INDR



Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B is used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written into the corresponding location of memory. Then HL and the byte counter are decremented. If decrementing causes B to go to zero, the instruction is terminated. If B is not zero, the PC is decremented by two and the instruction repeated. Note that if B is set to zero prior to instruction execution, 256 bytes of data will be input. Also, "Interrupts will be recognized and two refresh cycles will be executed after each data transfer".

If $B \neq 0$:

M CYCLES: 5 T STATES: 21(4,5,3,4,5) 4 MHZ E.T.: 5.25

If $B = 0$:

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S: Unknown
Z: Set
H: Unknown
P/V: Unknown
N: Set
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and the following sequence of bytes are available at the peripheral device mapped to I/O port address 07H:

51H
A9H
03H

then after the execution of

INDR

the HL register pair will contain 0FFDH, register B will contain zero, and memory locations will have contents as follows:

Location	Contents
0FFEh	03H
0FFFh	A9H
1000H	51H

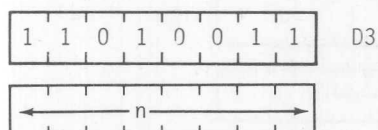
2.11.7

OUT (n), A

Operation: $(n) \leftarrow A$

Format:

<u>Opcode</u>	<u>Operands</u>
OUT	(n), A



Description:

The operand n is placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of the Accumulator (register A) also appear on the top half (A8 through A15) of the address bus at this time. Then the byte contained in the Accumulator is placed on the data bus and written into the selected peripheral device.

M CYCLES: 3 T STATES: 11(4,3,4) 4 MHZ E.T.: 2.75

Condition Bits Affected: None

Example:

If the contents of the Accumulator are 23H, then after the execution of

OUT 01H,A

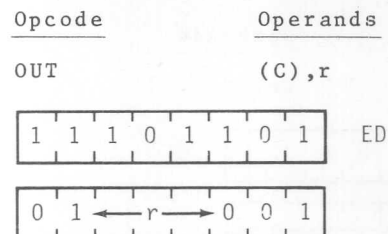
the byte 23H will have been written to the peripheral device mapped to I/O port address 01H.

2.11.8

OUT (C), r

Operation: (C) $\leftarrow$ r

Format:



Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of Register B are placed on the top half (A8 through A15) of the address bus at this time. Then the byte contained in register r is placed on the data bus and written into the selected peripheral device. Register r identifies any of the CPU registers shown in the following table, which also shows the corresponding 3-bit "r" field for each which appears in the assembled object code:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

M CYCLES: 3 T STATES: 12(4,4,4) 4 MHZ E.T.: 3.00

Condition Bits Affected: None

Example:

If the contents of register C are 01H and the contents of register D are 5AH, after the execution of

OUT (C),D

the byte 5AH will have been written to the peripheral device mapped to I/O port address 01H.

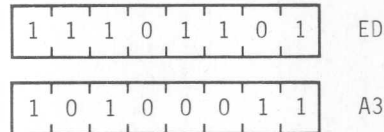
OUTI

Operation: $(C) \leftarrow (HL)$, $B \leftarrow B-1$, $HL \leftarrow HL+1$

Format:

Opcode

OUTI



Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus. The byte to be output is placed on the data bus and written into selected peripheral device. Finally the register pair HL is incremented.

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S: Unknown
 Z: Set if B-1=0;
 reset otherwise
 H: Unknown
 P/V: Unknown
 N: Set
 C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 10H, the contents of the HL register pair are 1000H, and the contents of memory address 1000H are

59H, then after the execution of

OUTI

register B will contain 0FH, the HL register pair will contain 1001H, and the byte 59H will have been written to the peripheral device mapped to I/O port address 07H.

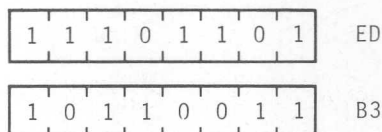
OTIR

Operation: $(C) \leftarrow (HL)$, $B \leftarrow B-1$, $HL \leftarrow HL + 1$

Format:

Opcode

OTIR



Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus at this time. Next the byte to be output is placed on the data bus and written into the selected peripheral device. Then register pair HL is incremented. If the decremented B register is not zero, the Program Counter (PC) is decremented by 2 and the instruction is repeated. If B has gone to zero, the instruction is terminated. Note that if B is set to zero prior to instruction execution, the instruction will output 256 bytes of data. Also, "Interrupts will be recognized and two refresh cycles will be executed after each data transfer".

If $B \neq 0$:

M CYCLES: 5 T STATES: 21(4,5,3,4,5) 4 MHZ E.T.: 5.25

If $B = 0$:

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S: Unknown
Z: Set
H: Unknown
P/V: Unknown
N: Set
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and memory locations have the following contents:

Location	Contents
1000H	51H
1001H	A9H
1002H	03H

then after the execution of

OTIR

the HL register pair will contain 1003H, register B will contain zero, and a group of bytes will have been written to the peripheral device mapped to I/O port address 07H in the following sequence:

51H
A9H
03H

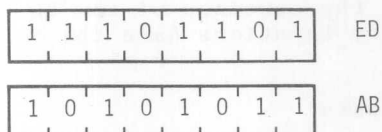
OUTD

Operation: $(C) \leftarrow (HL)$, $B \leftarrow B-1$, $HL \leftarrow HL-1$

Format:

Opcode

OUTD



Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus at this time. Next the byte to be output is placed on the data bus and written into the selected peripheral device. Finally the register pair HL is incremented.

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S: Unknown
 Z: Set if B-1=0;
 reset otherwise
 H: Unknown
 P/V: Unknown
 N: Set
 C: Not affected

Example:

If the contents of register C are 07H, the contents of

register B are 10H, the contents of the HL register pair are 1000H, and the contents of memory location 1000H are 59H, after the execution of

OUTD

register B will contain 0FH, the HL register pair will contain 0FFFH, and the byte 59H will have been written to the peripheral device mapped to I/O port address 07H.

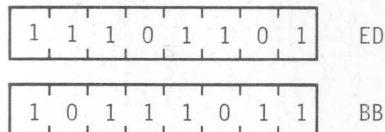
OTDR

Operation: $(C) \leftarrow (HL)$, $B \leftarrow B-1$, $HL \leftarrow HL-1$

Format:

Opcode

OTDR



Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus at this time. Next the byte to be output is placed on the data bus and written into the selected peripheral device. Then register pair HL is decremented and if the decremented B register is not zero, the Program Counter (PC) is decremented by 2 and the instruction is repeated. If B has gone to zero, the instruction is terminated. Note that if B is set to zero prior to instruction execution, the instruction will output 256 byte of data. Also, "Interrupts will be recognized and two refresh cycles will be executed after each data transfer".

If $B \neq 0$:

M CYCLES: 5 T STATES: 21(4,5,3,4,5) 4 MHZ E.T.: 5.25

If $B = 0$:

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S: Unknown
Z: Set
H: Unknown
P/V: Unknown
N: Set
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and memory locations have the following contents:

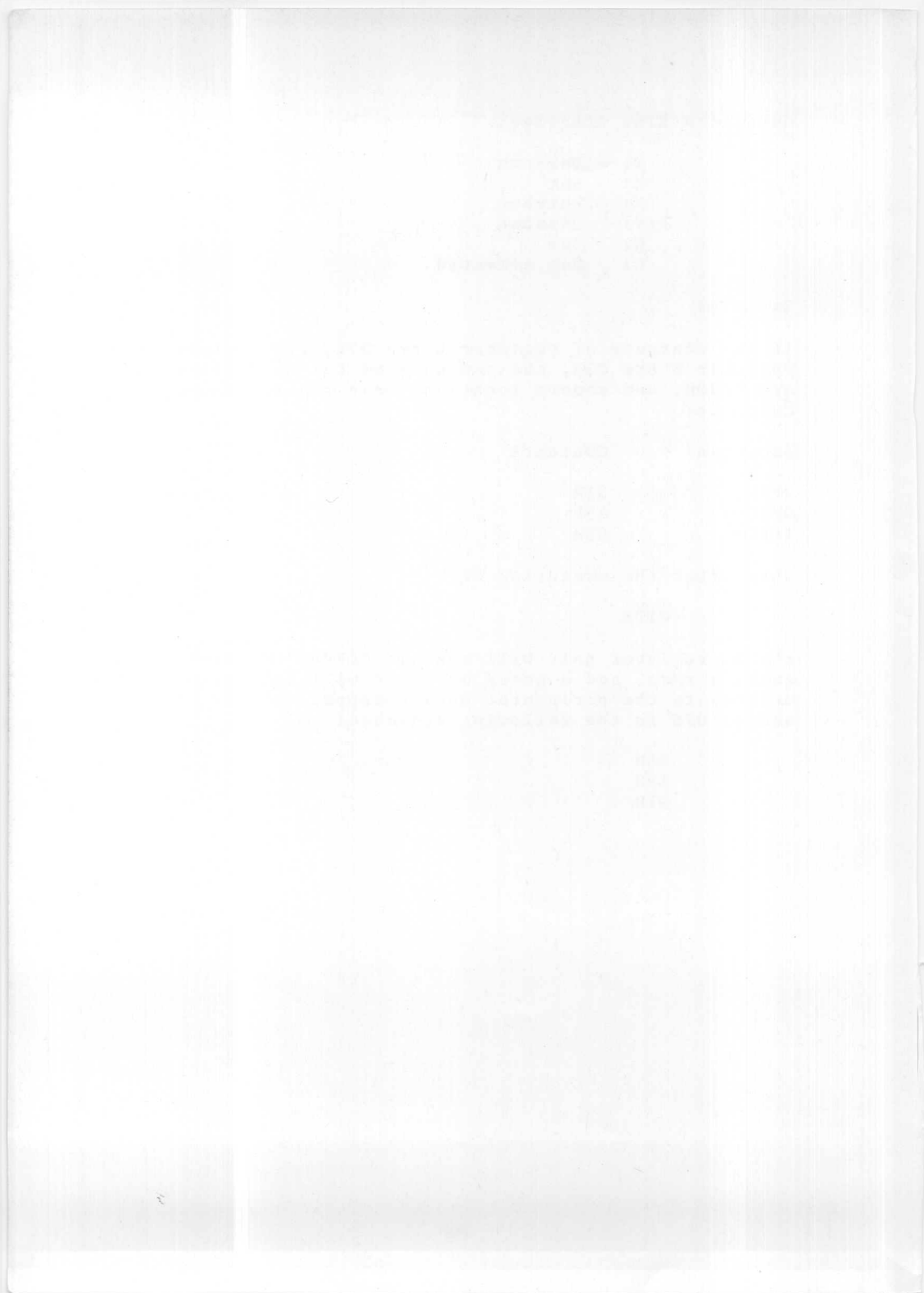
Location	Contents
OFFEH	51H
OFFFH	A9H
1000H	03H

then after the execution of

OTDR

the HL register pair will contain OFFDH, register B will contain zero, and a group of bytes will have been written to the peripheral device mapped to I/O port address 07H in the following sequence:

03H
A9H
51H



2.12 - FLAGS -

2.12.1 Z80 STATUS INDICATORS (FLAGS)

The flag register (F and F') supplies information to the user regarding the status of the Z80 at any given time. The bit positions for each flag is shown below:

7	6	5	4	3	2	1	0
S	Z	X	H	X	P/V	N	C

WHERE:

C = CARRY FLAG
N = ADD/SUBTRACT FLAG
P/V = PARITY/OVERFLOW FLAG
H = HALF-CARRY FLAG
Z = ZERO FLAG
S = SIGN FLAG
X = NOT USED

Each of the two Z-80 Flag Registers contains 6 bits of status information which are set or reset by CPU operations. (Bits 3 and 5 are not used.) Four of these bits are testable (C,P/V,Z and S) for use with conditional jump, call or return instructions. Two flags are not testable (H,N) and are used for BCD arithmetic.

CARRY FLAG (C)

The carry bit is set or reset depending on the operation being performed. For 'ADD' instructions that generate a carry and 'SUBTRACT' instructions that generate a borrow, the Carry Flag will be set. The Carry Flag is reset by an ADD that does not generate a carry and a 'SUBTRACT' that does not generate a borrow. This saved carry facilitates software routines for extended precision arithmetic. Also, the 'DAA' instruction will set the Carry Flag if the conditions for making the decimal adjustment are met.

For instructions RLA, RRA, RLS and RRS, the carry bit is used as a link between the LSB and MSB for any register or memory location. During instructions RLCA, RLC s and SLA s, the carry contains the last value shifted out of bit 7 of any register or memory location. During

instructions RRCA, RRC s, SRA s and SRL s the carry contains the last value shifted out of bit 0 of any register or memory location.

For the logical instructions AND s, OR s and XOR s, the carry will be reset.

The Carry Flag can also be set (SCF) and complemented (CCF).

ADD/SUBTRACT FLAG (N)

This flag is used by the decimal adjust accumulator instruction (DAA) to distinguish between 'ADD' and 'SUBTRACT' instructions. For all 'ADD' instructions, N will be set to an '0'. For all 'SUBTRACT' instructions, N will be set to a '1'.

PARITY/OVERFLOW FLAG

This flag is set to a particular state depending on the operation being performed.

For arithmetic operations, this flag indicates an overflow condition when the result in the Accumulator is greater than the maximum possible number (+127) or is less than the minimum possible number (-128). This overflow condition can be determined by examining the sign bits of the operands.

For addition, operands with different signs will never cause overflow. When adding operands with like signs and the result has a different sign, the overflow flag is set. For example:

+120 =	0111 1000	ADDEND
+105 =	0110 1001	AUGEND
+225	1110 0001	(-95) SUM

The two numbers added together has resulted in a number that exceeds +127 and the two positive operands has resulted in a negative number (-95) which is incorrect. The overflow flag is therefore set.

For subtraction, overflow can occur for operands of unlike signs. Operands of like sign will never cause overflow. For example:

+127	0111 1111	MINUEND
(-) -64	1100 0000	SUBTRAHEND
+191	1011 1111	DIFFERENCE

The minuend sign has changed from a positive to a negative, giving an incorrect difference. Overflow is therefore set.

Another method for predicting an overflow is to observe the carry into and out of the sign bit. If there is a carry in and no carry out, or if there is no carry in and a carry out, then overflow has occurred.

This flag is also used with logical operations and rotate instructions to indicate the parity of the result. The number of '1' bits in a byte are counted. If the total is odd, 'ODD' parity (P=0) is flagged. If the total is even, 'EVEN' parity is flagged (P=1).

During search instructions (CPI,CPIR,CPD,CPDR) and block transfer instructions (LDI,LDIR, LDD,LDDR) the P/V flag monitors the state of the byte count register (BC). When decrementing, the byte counter results in a zero value, the flag is reset to 0, otherwise the flag is a Logic 1.

During LD A,I and LD A,R instructions, the P/V flag will be set with the contents of the interrupt enable flip-flop (IFF2) for storage or testing.

When inputting a byte from an I/O device, IN r,(C), the flag will be adjusted to indicate the parity of the data.

THE HALF CARRY FLAG (H)

The Half Carry Flag (H) will be set or reset depending on the carry and borrow status between bits 3 and 4 of an 8-bit arithmetic operation. This flag is used by the decimal adjust accumulator instruction (DAA) to correct the result of a packed BCD add or subtract operation.

H	ADD	SUBTRACT
1	There is a carry from Bit 3 to Bit 4	There is a borrow from bit 4
0	There is no carry from Bit 3 to Bit 4	There is no borrow from Bit 4

THE ZERO FLAG (Z)

The Zero Flag (Z) is set or reset if the result generated by the execution of certain instructions is a zero.

For 8-bit arithmetic and logical operations, the Z flag will be set to a '1' if the resulting byte in the Accumulator is zero. If the byte is not zero, the Z flag is reset to '0'.

For compare (search) instructions, the Z flag will be set to a '1' if a comparison is found between the value in the Accumulator and the memory location pointed to by the contents of the register pair HL.

When testing a bit in a register or memory location, the Z flag will contain the complemented state of the indicated bit (see Bit b,s).

When inputting or outputting a byte between a memory location and an I/O device (INI;IND;OUTI and OUTD), if the result of B-1 is zero, the Z flag is set, otherwise it is reset. Also for byte inputs from I/O devices using IN r,(C), the Z Flag is set to indicate a zero byte input.

THE SIGN FLAG (S)

The Sign Flag (S) stores the state of the most significant bit of the Accumulator (Bit 7). When the Z80 performs arithmetic operations on signed numbers, binary two's complement notation is used to represent and process numeric information. A positive number is identified by a '0' in bit 7. A negative number is identified by a '1'. The binary equivalent of the magnitude of a positive number is stored in bits 0 to 6 for a total range of from 0 to 127. A negative number is represented by the two's complement of the equivalent positive number. The total range for negative numbers is from -1 to -128.

When inputting a byte from an I/O device to a register, IN r,(C), the S flag will indicate either positive (S=0) or negative (S=1) data.

INDEX

ALPHABETICAL ASSEMBLY MNEMONIC	OPERATION	PAGE	SECTION
ADC HL,ss	Add with Carry Reg. pair ss to HL.....	130	2- 6- 2
ADC A,s	Add with carry operand s to Acc.....	88	2- 4- 6
ADD A,n	Add value n to Acc.	82	2- 4- 2
ADD A,r	Add Reg. r to Acc.	80	2- 4- 1
ADD A,(HL)	Add location (HL) to Acc.	83	2- 4- 3
ADD A,(IX+d)	Add location (IX+d) to Acc.	84	2- 4- 4
ADD A,(IY+d)	Add location (IY+d) to Acc.	86	2- 4- 5
ADD HL,ss	Add Reg. pair ss to HL.....	128	2- 6- 1
ADD IX,pp	Add Reg. pair pp to IX.....	134	2- 6- 4
ADD IY,rr	Add Reg. pair rr to IY.....	136	2- 6- 5
AND s	Logical 'AND' of operand s and Acc.	94	2- 4- 9
BIT b,(HL)	Test BIT b of location (HL).....	188	2- 8- 2
BIT b,(IX+d)	Test BIT b of location (IX+d).....	190	2- 8- 3
BIT b,(IY+d)	Test BIT b of location (IY+d).....	192	2- 8- 4
BIT b,r	Test BIT b of Reg. r.....	186	2- 8- 1
CALL cc,nn	Call subroutine at location nn if condition cc is true.....	224	2-10- 2
CALL nn	Unconditional call subroutine at location nn.....	222	2-10- 1
CCF	Complement carry flag.....	117	2- 5- 4
CP s	Compare operand s with Acc.	100	2- 4-12
CPD	Compare location (HL) and Acc. decrement HL and BC.....	74	2- 3-13
CPDR	Compare location (HL) and Acc. decrement HL and BC, repeat until BC=0.....	76	2- 3-14
CPI	Compare location (HL) and Acc. increment HL and decrement BC.....	70	2- 3-11
CPIR	Compare location (HL) and Acc. increment HL, decrement BC repeat until BC=0.....	72	2- 3-12
CPL	Complement Acc. (1's comp.).....	114	2- 5- 2
DAA	Decimal adjust Acc.	112	2- 5- 1
DEC m	Decrement operand m.....	109	2- 4-17
DEC IX	Decrement IX.....	142	2- 6-10
DEC IY	Decrement IY.....	143	2- 6-11
DEC ss	Decrement Reg. pair ss.....	141	2- 6- 9
DI	Disable interrupts.....	121	2- 5- 8
DJNZ e	Decrement B and Jump relative if B=0.....	219	2- 9-11
EI	Enable interrupts.....	122	2- 5- 9
EX (SP),HL	Exchange the location (SP) and HL.....	59	2- 3- 4

		PAGE	SECTION
EX (SP),IX	Exchange the location (SP) and IX.....	60	2- 3- 5
EX (SP),IY	Exchange the location (SP) and IY.....	61	2- 3- 6
EX AF,AF'	Exchange the contents of AF and AF'.....	57	2- 3- 2
EX DE,HL	Exchange the contents of DE and HL.....	56	2- 3- 1
EXX	Exchange the contents of BC,DE,HL with contents of BC',DE',HL' respectively.....	58	2- 3- 3
HALT	HALT (wait for interrupt or reset).....	120	2- 5- 7
IM 0	Set interrupt mode 0.....	123	2- 5-10
IM 1	Set interrupt mode 1.....	124	2- 5-11
IM 2	Set interrupt mode 2.....	125	2- 5-12
IN A,(n)	Load the Acc. with input from device n.....	238	2-11- 1
IN r,(C)	Load the Reg. r with input from device (C).....	239	2-11- 2
INC (HL)	Increment location (HL).....	104	2- 4-14
INC IX	Increment IX.....	139	2- 6- 7
INC (IX+d)	Increment location (IX+d).....	105	2- 4-15
INC IY	Increment IY.....	140	2- 6- 8
INC (IY+d)	Increment location (IY+d).....	107	2- 4-16
INC r	Increment Reg. r.....	102	2- 4-13
INC ss	Increment Reg. pair ss.....	138	2- 6- 6
IND	Load location (HL) with input from port (C), decrement HL and B.....	245	2-11- 5
INDR	Load location (HL) with input from port (C), decrement HL and decrement B, repeat until B=0.....	247	2-11- 6
INI	Load location (HL) with input from port (C); or increment HL and decrement B.....	241	2-11- 3
INIR	Load location (HL) with input from port (C), increment HL and decrement B, repeat until B=0.....	243	2-11- 4
JP (HL)	Unconditional Jump to (HL).....	216	2- 9- 8
JP (IX)	Unconditional Jump to (IX).....	217	2- 9- 9
JP (IY)	Unconditional Jump to (IY).....	218	2- 9-10
JP cc,nn	Jump to location nn if condition cc is true.....	205	2- 9- 2
JP nn	Unconditional jump to location nn.....	204	2- 9- 1
JK C,e	Jump relative to PC+e if carry=1.....	208	2- 9- 4
JK e	Unconditional Jump relative to PC+e.....	207	2- 9- 3
JK NC,e	Jump relative to PC+e if carry=0.....	210	2- 9- 5

		PAGE	SECTION
JR NZ,e	Jump relative to PC+e if non zero (Z=0).....	214	2- 9- 7
JR Z,e	Jump relative to PC+e if zero (Z=1).....	212	2- 9- 6
LD A,(BC)	Load Acc. with location (BC).....	21	2- 1-12
LD A,(DE)	Load Acc. with location (DE).....	22	2- 1-13
LD A,I	Load Acc. with I.....	27	2- 1-18
LD A,(nn)	Load Acc. with location nn.....	23	2- 1-14
LD A,R	Load Acc. with Reg. R.....	28	2- 1-19
LD (BC),A	Load location (BC) with Acc.	24	2- 1-15
LD (DE),A	Load location (DE) with Acc.	25	2- 1-16
LD (HL),n	Load location (HL) with value n.....	18	2- 1- 9
LD dd,nn	Load Reg. pair dd with value nn.....	32	2- 2- 1
LD dd,(nn)	Load Reg. pair dd with location (nn)....	36	2- 2- 5
LD HL,(nn)	Load HL with location (nn).....	35	2- 2- 4
LD (HL),r	Load location (HL) with Reg. r.....	13	2- 1- 6
LD I,A	Load I with Acc.	29	2- 1-20
LD IX,nn	Load IX with value nn.....	33	2- 2- 2
LD IX,(nn)	Load IX with location (nn).....	38	2- 2- 6
LD (IX+d),n	Load location (IX+d) with value n.....	19	2- 1-10
LD (IX+d),r	Load location (IX+d) with Reg. r.....	14	2- 1- 7
LD IY,nn	Load IY with value nn.....	34	2- 2- 3
LD IY,(nn)	Load IY with location (nn).....	39	2- 2- 7
LD (IY+d),n	Load location (IY+d) with value n.....	20	2- 1-11
LD (IY+d),r	Load location (IY+d) with Reg. r.....	16	2- 1- 8
LD (nn),A	Load location (nn) with Acc.	26	2- 1-17
LD (nn),dd	Load location (nn) with Reg. pair dd....	41	2- 2- 9
LD (nn),HL	Load location (nn) with HL.....	40	2- 2- 8
LD (nn),IX	Load location (nn) with IX.....	43	2- 2-10
LD (nn),IY	Load location (nn) with IY.....	44	2- 2-11
LD R,A	Load R with Acc.	30	2- 1-21
LD r,(HL)	Load Reg. r with location (HL).....	8	2- 1- 3
LD r,(IX+d)	Load Reg. r with location (IX+d).....	9	2- 1- 4
LD r,(IY+d)	Load Reg. r with location (IY+d).....	11	2- 1- 5
LD r,n	Load Reg. r with value n.....	7	2- 1- 2
LD r,r'	Load Reg. r with Reg. r'.....	6	2- 1- 1
LD SP,HL	Load SP with HL.....	45	2- 2-12
LD SP,IX	Load SP with IX.....	46	2- 2-13
LD SP,IY	Load SP with IY.....	47	2- 2-14
LDD	Load location (DE) with location (HL), decrement DE,HL and BC.....	66	2- 3- 9
LDDR	Load location (DE) with location (HL), decrement DE,HL and BC; repeat until BC=0.....	68	2- 3-10

		PAGE	SECTION
LDI	Load location (DE) with location (HL), increment DE,HL, decrement BC.....	62	2- 3- 7
LDIR	Load location (DE) with location (HL), increment DE,HL, decrement BC and repeat until BC=0.....	64	2- 3- 8
NEG	Negate Acc. (2's complement).....	115	2- 5- 3
NOP	No operation.....	119	2- 5- 6
OR s	Logical 'OR' of operand s and Acc.	96	2- 4-10
OTDR	Load output port (C) with location (HL) decrement HL and B, repeat until B=0.....	258	2-11-12
OTIR	Load output port (C) with location (HL), increment HL, decrement B, repeat until B=0.....	254	2-11-10
OUT (C),r	Load output port (C) with Reg. r.....	250	2-11- 8
OUT (n),A	Load output port (n) with Acc.	249	2-11- 7
OUTD	Load output port (C) with location (HL), decrement HL and B.....	256	2-11-11
OUTI	Load output port (C) with location (HL), increment HL and decrement B.....	252	2-11- 9
POP IX	Load IX with top of stack.....	53	2- 2-19
POP IY	Load IY with top of stack.....	54	2- 2-20
POP qq	Load Reg. pair qq with top of stack.....	51	2- 2-18
PUSH IX	Load IX onto stack.....	49	2- 2-16
PUSH IY	Load IY onto stack.....	50	2- 2-17
PUSH qq	Load Reg. pair qq onto stack.....	48	2- 2-15
RES b,m	Reset Bit b of operand m.....	200	2- 8- 9
RET	Return from subroutine.....	227	2-10- 3
RET cc	Return from subroutine if condition cc is true.....	228	2-10- 4
RETI	Return from interrupt.....	230	2-10- 5
RETN	Return from non maskable interrupt.....	232	2-10- 6
RL m	Rotate left through carry operand m.....	162	2- 7- 9
RLA	Rotate left Acc. through carry.....	148	2- 7- 2
RLC (HL)	Rotate location (HL) left circular.....	156	2- 7- 6
RLC (IX+d)	Rotate location (IX+d) left circular....	158	2- 7- 7
RLC (IY+d)	Rotate location (IY+d) left circular....	160	2- 7- 8
RLC r	Rotate Reg. r left circular.....	154	2- 7- 5
RLCA	Rotate left circular Acc.	146	2- 7- 1
RLD	Rotate digit left and right between Acc. and location (HL).....	180	2- 7-15
RR m	Rotate right through carry operand m....	168	2- 7-11
RRA	Rotate right Acc. through carry.....	152	2- 7- 4
RRC m	Rotate operand m right circular.....	165	2- 7-10

		PAGE	SECTION
RRCA	Rotate right circular Acc.	150	2- 7- 3
RRD	Rotate digit right and left between Acc. and location (HL).....	182	2- 7-16
RST p	Restart to location p.....	234	2-10- 7
SBC A,s	Subtract operand s from Acc. with carry.....	92	2- 4- 8
SBC HL,ss	Subtract Reg. pair ss from HL with carry.....	132	2- 6- 3
SCF	Set carry flag (C=1).....	118	2- 5- 5
SET b,(HL)	Set Bit b of location (HL).....	195	2- 8- 6
SET b,(IX+d)	Set Bit b of location (IX+d).....	196	2- 8- 7
SET b,(IY+d)	Set Bit b of location (IY+d).....	198	2- 8- 8
SET b,r	Set Bit b of Reg. r.....	194	2- 8- 5
SLA m	Shift operand m left arithmetic.....	171	2- 7-12
SRA m	Shift operand m right arithmetic.....	174	2- 7-13
SRL m	Shift operand m right logical.....	177	2- 7-14
SUB s	Subtract operand s from Acc.	90	2- 4- 7
XOR s	Exclusive 'OR' operand s and Acc.	98	2- 4-11

Information furnished is believed to be accurate and reliable. However, no responsibility is assumed for the consequences of its use nor for an infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of SGS-ATES. This publication supersedes and substitutes all information previously supplied.

SGS - ATES GROUP OF COMPANIES

Italy - France - Germany - Singapore - Sweden - United Kingdom - U.S.A.

Printed in Italy by permission of Zilog Inc.

SGS-ATES GROUP OF COMPANIES

INTERNATIONAL HEADQUARTERS

SGS-ATES Componenti Elettronici SpA
Via C. Olivetti 2 - 20041 Agrate Brianza - Italy
Tel.: 039-650341÷4/650441÷5/650841÷5
Telex: 36141-36131

BENELUX

SGS-ATES Componenti Elettronici SpA
Benelux Sales Office
-B-1180 Bruxelles
Winston Churchill Avenue, 122
Tel.: 02-3432439
Telex: 24149 B

DENMARK

SGS-ATES Scandinavia AB
Sales Office:
2730 Herlev
Marielundvej 46D
Tel.: 02-948533
Telex: 35280

FRANCE

SGS-ATES France S.A.
75643 Paris Cedex 13
Résidence "Le Palatino"
17, Avenue de Choisy
Tel.: 5842730
Telex: 021-25938

GERMANY

SGS-ATES Deutschland Halbleiter
Bauelemente GmbH
8018 Grafting bei München
Haidling 17
Tel.: 08092-691
Telex: 032-527370
Sales Offices:
1000 Berlin 20
Gatower Strasse 185
Tel.: 030-3622031
Telex: 01 85418
3000 Hannover 1
Lange Laube 19
Tel.: 0511-17522/3
Telex: 09 23195
8000 München 40
Gernotstrasse 10
Tel.: 089-304270/304485
Telex: 05 215784
8500 Nurnberg 15
Parsifalstrasse 10
Tel.: 0911-40645
7000 Stuttgart 80
Kalifenweg 45
Tel.: 0711-713091/2
Telex: 07 255545

ITALY

SGS-ATES Componenti Elettronici SpA
Sales Offices:
50127 Firenze
Via Giovanni Del Pian Dei Carpi 96/1
Tel.: 055-4377763
20149 Milano
Via Correggio 1/3
Tel.: 02-4695651
00199 Roma
Piazza Gondar 11
Tel.: 06-8392848/8312777
10134 Torino
Via La Loggia 51/7
Tel.: 011-634572

NORWAY

SGS-ATES Scandinavia AB
Sales Office:
Oslo 9
Haavard Martinsens Vei 19
Tel.: 10 60 50
Telex: 11796

SINGAPORE

SGS-ATES Singapore (Pte) Ltd.
Singapore 12
Lorong 4 & 6 - Toa Payoh
Tel.: 531411
Telex: ESGIES RS 21412

SWEDEN

SGS-ATES Scandinavia AB
19501 Märsta
Tingvallavaegen 9J
Tel.: 0760-40120
Telex: 042-10932

UNITED KINGDOM

SGS-ATES (United Kingdom) Ltd.
Aylesbury, Bucks
Planar House, Walton Street
Tel.: 0296-5977
Telex: 041-83245